A PROJECT REPORT

ON

"Human Activity Identification"

SUBMITTED TO
THE SAVITRIBAI PHULE PUNE UNIVERSITY, PUNE
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE AWARD OF THE DEGREE

BACHELOR OF ENGINEERING

In

COMPUTER ENGINEERING

of

SAVITRIBAI PHULE PUNE UNIVERSITY

By

NEHA MARNE	B150234313
AKANSHA JAGTAP	B150234277
ISSHITA PALIWAL	B150234272

Under the guidance of

Prof. M. P. Wankhade

Sinhgad Institutes

DEPARTMENT OF COMPUTER ENGINEERING
SINHGAD COLLEGE OF ENGINEERING, PUNE-41
Accredited by NAAC with grade 'A'

YEAR 2021-22

Sinhgad Technical Education Society,

Department of Computer Engineering

Sinhgad College of Engineering, Pune-41

Accredited by NAAC with grade 'A'

Date:

CERTIFICATE

This is to certify that this project report entitled

"Human Activity Identification"

Submitted by

NEHA MARNE	B150234313
AKANSHA JAGTAP	B150234277
ISSHITA PALIWAL	B150234272

is a bonafide work carried out by them under the supervision of Prof. M. P. Wankhade and it is approved for the partial fulfillment of the requirements of Savitribai Phule Pune University , Pune for the award of the degree of Bachelor of Engineering (Computer Engineering) during the year 2021-22.

Prof. M. P. Wankhade
Internal Guide
Department of Computer Engineering

Prof. M. P. Wankhade
Head of Dept.
Department of Computer Engineering

Dr. S.D. Lokhande
Principal
SCOE, Pune

ACKNOWLEDGEMENT

Every orientation works has an imprint of many people and it becomes our duty to express deep gratitude for the same.

During the entire duration of preparation for this Dissertation, we received endless help from a number of people and feel that this report would be incomplete if we do not convey graceful thanks to them. This acknowledgement is a humble attempt to thanks all those who were involved in the project work and were of immense help to us.

First and foremost we take the opportunity to extend our deep heartfelt gratitude to our guide and **Head of Department of Computer Engineering SCOE, Pune Prof M. P. Wankhade** for guiding us throughout the entire project and for his kind and valuable suggestions, without which this idea won't have executed. We also humbly thank for his indispensable support, his priceless suggestions and for his valuable time.

ABSTRACT

Human activity recognition, or HAR for short, is a broad field of study concerned with identifying the specific movement or action of a person based on sensor data.

Movements are often typical activities performed indoors, such as walking, talking, standing, and sitting. They may also be more focused activities such as those types of activities performed in a kitchen or on a factory floor. It will mainly be used for eldercare and healthcare as an assistive technology when ensemble with other technologies likes Internet of Things (IoT). HAR can be done with the help of sensors, smartphones or images. Activity recognition is used in many applications such as surveillance, anti-terrorists, and anti-crime securities as well as life logging and assistance.

Human activity recognition plays a significant role in human-to- human interaction and interpersonal relations. The human ability to recognize another person's activities is one of the main subjects of study of the scientific areas of computer vision and machine learning. With the advent of Pose estimation and classification algorithm, which can be used on images/video input, it is now possible to collect and store data on different aspects of human movement under the conditions of free living.This technology has the potential to be used in automated activity profiling systems which produce a continuous record of activity patterns over extended periods of time. Such activity profiling systems are dependent on classification algorithms which can effectively interpret motion data and identify different activities. This report reviews the different techniques which have been used to classify normal activities and/or identify falls from body-joints data. The report is structured according to the different analytical techniques and illustrates the variety of approaches which have previously been applied in this field. Although significant progress has been made in this important area, there is still significant scope for further work, particularly in the application of advanced classification techniques to problems involving many different activities.

This project will analyze the activity being performed by the user in the Video/Image input. Human activity recognition will use Pose estimation and classification algorithm to analyze the data set and detect the activity.

LIST OF FIGURES

Figure 1.1 MediaPipe....6
Figure 1.2 Pose Estimation Key Points....6
Figure 1.3 Human Pose Estimation Pipeline....7
Figure 1.4 Vitruvian man aligned via two virtual keypoints predicted by BlazePose detector in addition to the face bounding box....7
Figure 1.5 Tracking network architecture: regression with heatmap supervision....8
Figure 1.6 Multinomial Logistic Regression....10
Figure 2.1 Agile Process Model....14
Figure 3.1 System Workflow....17
Figure 3.2 System Architecture....18
Figure 3.3 Use Case Diagram....19
Figure 3.4 Class diagram....20
Figure 3.5 Activity Diagram....21
Figure 3.6 Sequence Diagram....22
Figure 3.7 Data Flow(0) diagram....23
Figure 3.8 Data Flow(1) diagram....23
Figure 3.9 Data Flow(2) diagram....23
Figure 4.1 Flowchart of system....44
Figure 5.1 Unit Testing 1....45
Figure 5.2 Unit Testing 2....46
Figure 5.3 Functional Testing 1....46
Figure 5.4 Functional Testing 2....47
Figure 6.1 Confusion Matrix 1....50
Figure 6.2 Confusion Matrix 2....51
Figure 6.3 Confusion Matrix 3....51
Figure 6.4 Dashboard....52
Figure 6.5 Action Identification model....53
Figure 6.6 Help Tab....53
Figure 6.7 Upload Videos....54
Figure 6.8 View all uploaded videos....54
Figure 6.9 Action Detected....55
Figure 6.10 Action Detected Standing....55

Figure 6.11 No Action Detected 56
Figure 6.12 Live video Input (Standing) 56
Figure 6.13 Live video Input (Namaste) 57
Figure 6.14 Live video Input (Left hand up) 57
Figure 6.15 Live video Input (Right hand up) 58
Figure 6.16 Live video Input (Bend down) 58
Figure 6.17 Live video input (Hands on Waist) 59
Figure 6.18 Live video input (no human identified) 59
Figure 6.19 Multiple Person Activity Identification 60

LIST OF TABLES

Table 1.1 Literature Survey Summary....4
Table 1.2 Dataset format....9
Table 1.3 Actions in dataset....9
Table 2.1 Cost Estimate Table....15
Table 2.2 Timeline Chart....16
Table 3.1 I Elements....24
Table 3.2 D Elements....24
Table 3.3 E Elements....24
Table 3.4 A Elements....24
Table 5.1 Manual Testing Test Cases....47

INDEX

Contents

TITLE PAGE i

CERTIFICATE ii

LIST OF FIGURES v

LIST OF TABLES vii

INDEX viii

1: INTRODUCTION 1

1.1 Background and Basics: 1

1.2 Literature Survey: 1

1.3 Project Undertaken: 4

1.3.1 Problem Definition: 4

1.3.2 Scope Statement: 5

1.3.3 Motivation: 5

1.3.4 Objective: 5

1.3.5 Proposed System: 5

1.4 Organization of Project Report: 11

2: PROJECT PLANNING AND MANAGEMENT 12

2.1 Introduction: 12

2.2 Detailed System Requirement Specification: 12

2.2.1 System Overview: 12

2.2.2 Non Functional Requirements: 12

2.2.3 Software And Hardware Requirements: 13

2.3 Process Model: 14

2.4 Cost and Effort Estimates: 15

2.5 Project Scheduling: 16

3: ANALYSIS AND DESIGN 17

3.1 Introduction: 17

3.2 System Architecture: 17
3.3 Use Case Diagram: 19
3.4 Class Diagram: 20
3.5 Activity Diagram: 21
3.6 Sequence Diagram: 22
3.7 Data Flow Diagram: 23
3.8 IDEA Matrix: 24
3.9 Mathematical Model: 25
4: IMPLEMENTATION AND CODING 26
4.1 Introduction: 26
4.2 Implementation: 26
5: TESTING 45
5.1 Introduction: 45
5.2 Unit Testing: 45
5.3 Functional Testing: 46
5.4 Manual Testing: 47
6: RESULTS AND DISCUSSIONS 50
6.1 Result: 50
6.2 Main GUI Snapshots: 52
6.3 Future Scope: 60
7: CONCLUSION 61
8: REFERENCES 62

1: INTRODUCTION

1.1 Background and Basics:

Human activity recognition has a wide range of uses because of its impact on wellbeing. Human activities have an inherent hierarchical structure that indicates the different levels of it, which can be considered as a three-level categorization. Movements are often typical activities performed indoors, such as walking, talking, standing, andsitting. They may also be more focused activities such as those types of activities performed in a kitchen or on a factory floor. It will mainly be used for eldercare and healthcare as an assistive technology when ensemble with other technologies likes Internet of Things (IoT). HAR can be done with the help of sensors, smartphones or images. Activity recognition is used in many applications such as surveillance, anti-terrorists, and anti-crime securities as well as life logging and assistance.

1.2 Literature Survey:

After studying various work related Human Activity Recognition we've summarized it as follows:

Gavrila (1999) separated the research in 2D (with and without explicit shape models) and 3D approaches. In Aggarwal and Cai (1999), a new taxonomy was presented focusing on human motion analysis, tracking from single view and multiview cameras, and recognition of human activities. Similar in spirit to the previous taxonomy, Wang et al. (2003) proposed a hierarchical action categorization hierarchy. The survey of Moeslund et al. (2006) mainly focused on pose-based action recognition methods and proposed a fourfold taxonomy, including initialization of human motion, tracking, pose estimation, and recognition methods.

A fine separation between the meanings of "action" and "activity" was proposed by Turaga et al. (2008), where the activity recognition methods were categorized according to their degree of activity complexity. Poppe (2010) characterized human activity recognition methods into two main categories, describing them as "top-down" and "bottom-up." On the other hand, Aggarwal and Ryoo (2011) presented a tree-structured taxonomy, where the human activity recognition methods were categorized into two big sub-categories, the "single layer" approaches and the "hierarchical" approaches, each of which have several layers of categorization.

Modeling 3D data is also a new trend, and it was extensively studied by Chen et al. (2013b) and Ye et al. (2013). As the human body consists of limbs connected with joints, one can model these parts using stronger features, which are obtained from depth cameras, and create a 3D representation of the human body, which is more informative than the analysis of 2D activities carried out in the image plane. Aggarwal and Xia (2014) recently presented a categorization of human activity recognition methods from 3D stereo and motion capture systems with the main focus on methods that exploit 3D depth data. To this end, Microsoft Kinect has played a significant role in motion capture of articulated body skeletons using depth sensors.

Although much research has been focused on human activity recognition systems from video sequences, human activity recognition from static images remains an open and very challenging task. Most of the studies of human activity recognition are associated with facial expression recognition and/or pose estimation techniques. Guo and Lai (2014) summarized all the methods for human activity recognition from still images and categorized them into two big categories according to the level of abstraction and the type of features each method uses.

Jaimes and Sebe (2007) proposed a survey for multimodal human computer interaction focusing on affective interaction methods from poses, facial expressions, and speech. Pantic and Rothkrantz (2003) performed a complete study in human affective state recognition methods that incorporate non-verbal multimodal cues, such as facial and vocal expressions. Pantic et al. (2006) studied several state-of-the-art methods of human behavior recognition including affective and social cues and covered many open computational problems and how they can be efficiently incorporated into a human-computer interaction system.

Zeng et al. (2009) presented a review of state-of-the-art affective recognition methods that use visual and audio cues for recognizing spontaneous affective states and provided a list of related datasets for human affective expression recognition. Bousmalis et al. (2013a) proposed an analysis of non-verbal multimodal (i.e., visual and auditory cues) behavior recognition methods and datasets for spontaneous agreements and disagreements. Such social attributes may play an important role in analyzing social behaviors, which are the key to social engagement. Finally, a through analysis of the technologies for human behavior recognition from the viewpoint of data and knowledge representation was presented by Rodríguez et al. (2014).

Table 1.1 Literature Survey Summary

Sr.no	Title	Author	Abstract
1	Human motion analysis: A review.	Aggarwal and Cai (1999)	Human motion analysis, tracking from single view and multiview cameras, and recognition of human activities.
2	A survey of advances in vision-based human motion capture and analysis	Moeslund et al. (2006)	Mainly focused on pose-based action recognition methods and proposed a fourfold taxonomy, including initialization of human motion, tracking, pose estimation, and recognition methods.
3	Deep Convolutional Neural Networks for Human Action Recognition Using Depth Maps and Postures	Aouaidjia Kamel, Bin Sheng, Po Yang, Ping L, Ruimin Shen, David Dagan Feng(2018)	Presented a method (Action-Fusion) for human action recognition from depth maps and posture data using convolutional neural networks (CNNs).
4	Exploiting temporal information for 3D human pose estimation	Mir Rayat Imtiaz Hossain, James J. Little(2018)	Utilize the temporal information across a sequence of 2D joint locations to estimate a sequence of 3D poses. We designed a sequence-to-sequence network composed of layer-normalized LSTM units with shortcut connections connecting the input to the output on the decoder side and imposed temporal smoothness constraint during training.
5	Efficient Frequency Domain Feature Extraction Model using EPS and LDA for Human Activity Recognition	Rasel Ahmed Bhuiyan, Nadeem Ahmed (2020)	Enveloped Power Spectrum (EPS) is used for extracting impulse components of the signal, and the Linear Discriminant Analysis (LDA) is used as a dimensionality reduction procedure to extract the discriminant features for

			human daily activity recognition. After completing EPS feature extraction techniques, LDA is performed on those extracted spectra for extracting features using the dimension reduction technique. Finally, the discriminant vocabulary vector is trained by the Multiclass Support Vector Machine (MCSVM) to classify human activities
6	Human Activity Recognition Using Pose Estimation and Machine Learning Algorithm	Abhay Gupta, Kuldeep Gupta, Kshama Gupta and Kapil Gupta(2021)	A single person poses estimation and activity classification using pose. Pose Estimation consists of the recognition of 18 body key points and joints locations. We have used the OpenPose library for pose estimation work. And the activity classification task is performed by using multiple logistic regression.

After doing this survey we've analyzed that most of the human action identification techniques use sensors or such complicated and expensive techniques and so the sensors are hard to carry. The pose estimation technique which we have discussed is easier to implement and cheap

1.3 Project Undertaken:

The aim is to build a software that will recognize human activities. We can also predict actions using already recoded videos as well as using live footage.

1.3.1 Problem Definition:

Human Activity Recognition is the problem of predicting what a person is doing based on a trace of their movement using Pose estimation and Classification algorithm. It is a challenging problem because there are many motions involved to specific actions in a general way. Human activity recognition is a field of study that deals with identifying, interpreting, and analysing the actions specific to the movement of human beings.

1.3.2 Scope Statement:

This project will analyze the activity being performed by the user in the Video/Image input. Human activity recognition will use Pose estimation and classification algorithm to analyze the data set and detect the activity.

1.3.3 Motivation:

- Human activity recognition basis for many applications such as video surveillance, health care, and human-computer interaction. To analyze the activity of a person from the information collected by different devices.
- Discover which are the variables that determine the activity.
- To calculate a predictive model that can recognize a person's activity from the signals received by the sensors.
- Design individualized exercise tables to improve the health of a person.

1.3.4 Objective:

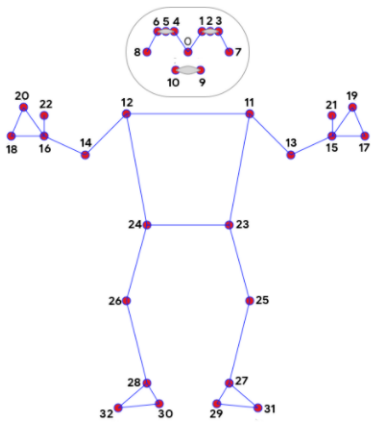
- To identify a method achieving more accurate human activity recognition by using suitable tools.
- To propose a mechanism of an automated analysis or interpretation of ongoing events and their context from video data.
- To explore better approach for action recognition based on using the suitable types of data to balance the use of features by strengthen the weak part in eachtype by the strong part in the other. To recommend effective methods for the most cost-effective human activity recognition.

1.3.5 Proposed System:

Pose estimation:

Pose Estimation is a computer vision task that infers the pose of a person or object in an image or video. Pose estimation is the problem of determining the position and orientation of a camera relative to a given person or object. This is typically done by identifying, locating, and tracking a number of key points on a given object or person. For objects, this could be corners or other significant features. And for humans, these keypoints represent major joints like an elbow or knee. The goal of our machine learning models is to track these keypoints in images and videos.

Figure 1.1 MediaPipe



0. nose
1. left_eye_inner
2. left_eye
3. left_eye_outer
4. right_eye_inner
5. right_eye
6. right_eye_outer
7. left_ear
8. right_ear
9. mouth_left
10. mouth_right
11. left_shoulder
12. right_shoulder
13. left_elbow
14. right_elbow
15. left_wrist
16. right_wrist
17. left_pinky
18. right_pinky
19. left_index
20. right_index
21. left_thumb
22. right_thumb
23. left_hip
24. right_hip
25. left_knee
26. right_knee
27. left_ankle
28. right_ankle
29. left_heel
30. right_heel
31. left_foot_index
32. right_foot_index

Figure 1.2 Pose Estimation Key Points

An ML Pipeline for Pose Tracking:

For pose estimation, we utilize our proven two-step detector-tracker ML pipeline. Using a detector, this pipeline first locates the pose region-of-interest (ROI) within the frame. The tracker subsequently predicts all 33 pose keypoints from this ROI. Note that for video use cases, the detector is run only on the first frame. For subsequent frames we derive the ROI from the previous frame's pose.

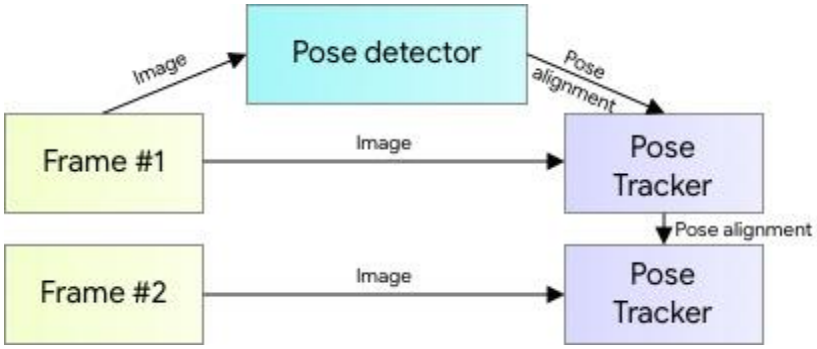


Figure 1.3 Human Pose Estimation Pipeline

Person/pose Detection Model (BlazePose Detector):

The detector is inspired by our own lightweight BlazeFace model, used in MediaPipe Face Detection, as a proxy for a person detector. It explicitly predicts two additional virtual keypoints that firmly describe the human body center, rotation and scale as a circle. Inspired by Leonardo's Vitruvian man, we predict the midpoint of a person's hips, the radius of a circle circumscribing the whole person, and the incline angle of the line connecting the shoulder and hip midpoints.

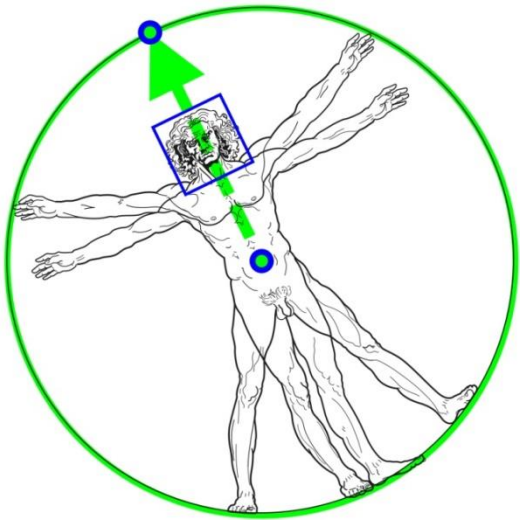


Figure 1.4 Vitruvian man aligned via two virtual keypoints predicted by BlazePose detector in addition to the face bounding box.

Tracking Model

The pose estimation component of the pipeline predicts the location of all 33 person keypoints with three degrees of freedom each (*x*, *y* location and visibility) plus the two virtual alignment keypoints described above. Unlike current approaches that employ compute-intensive heatmap prediction, our model uses a regression approach that is *supervised* by a combined heat map/offset prediction of all keypoints, as shown below.

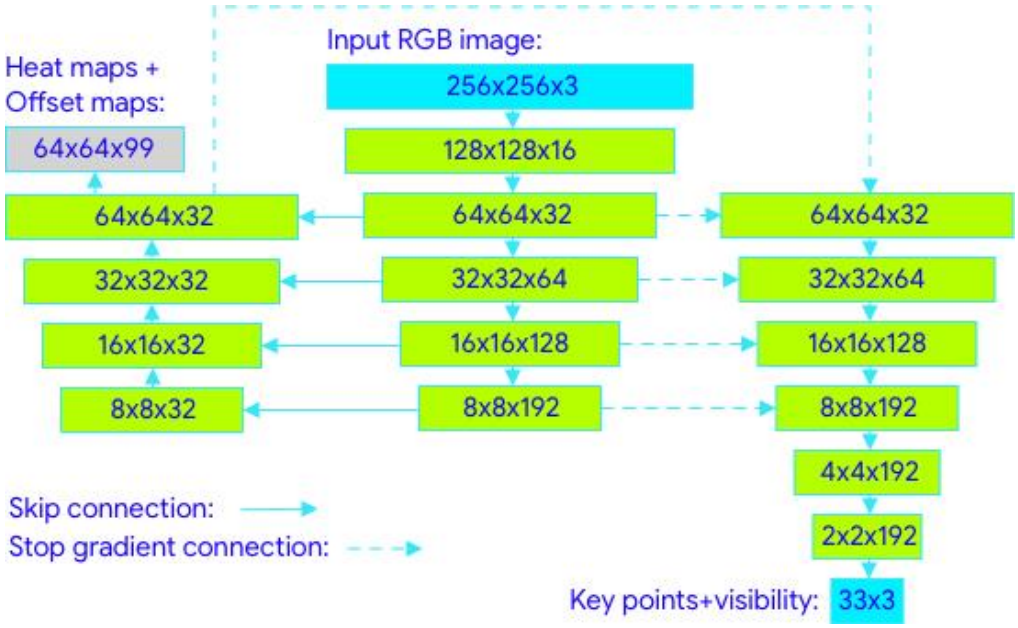


Figure 1.5 Tracking network architecture: regression with heatmap supervision

x and y - Landmark coordinates normalized to [0.0, 1.0] by the image width and height respectively. z: Represents the landmark depth with the depth at the midpoint of hips being the origin, and the smaller the value the closer the landmark is to the camera. The magnitude of z uses roughly the same scale as x. Visibility: A value in [0.0, 1.0] indicating the likelihood of the landmark being visible (present and not occluded) in the image.

Dataset Creation:

Based on the mediapipe analysis we create our CSV file to train our Classification Algorithm and Prediction is made about the human activity. The admin can use webcam for live input or upload video input and according to the details the prediction will be made. Logistic Regression was used to predict the action.

Table 1.2 Dataset format

	class	x1	y1	z1	v1
0	Standing	0.441929	0.179151	-0.768026	0.999997
1	Standing	0.441217	0.179151	-0.949387	0.999997
2	Standing	0.440749	0.179168	-0.946997	0.999997
3	Standing	0.440405	0.179345	-0.940684	0.999997
4	Standing	0.440183	0.179372	-0.959358	0.999996

Table 1.3 Actions in dataset

Sr No.	Actions	Numbers of entries
1	Namaste	468
2	Standing	424
3	Hands_On_Waist	420
4	Bend_Down	331
5	Left_Hand_Up	272
6	Right_Hand_Up	252

Classification Algorithm: Multinomial Logistic Regression:

- Multinomial logistic regression is used to predict categorical placement in or the probability of category membership on a dependent variable based on multiple independent variables.
- The independent variables can be either dichotomous (i.e., binary) or continuous (i.e., interval or ratio in scale).
- Multinomial logistic regression is a simple extension of binary logistic regression that allows for more than two categories of the dependent or outcome variable.

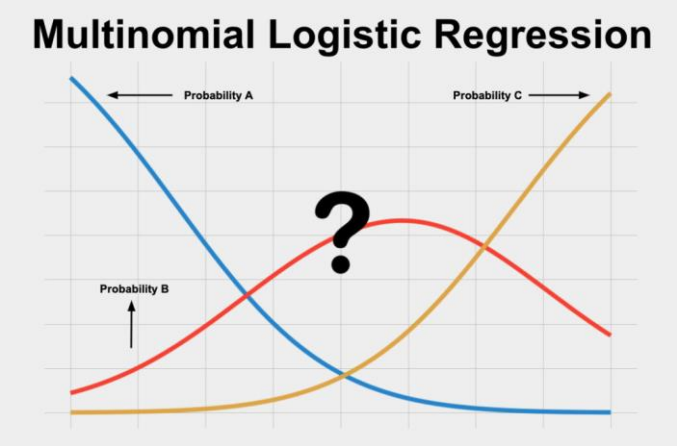


Figure 1.6 Multinomial Logistic Regression

- Advantages:
  - Helps to understand the relationships among the variables present in the dataset.
  - Simultaneous Models result in smaller standard errors for the parameter estimates than when fitting the logistic regression models separately.
  - The choice of reference class has no effect on the parameter estimates for other categories.

$$\Pr(Y_i = 1) = \frac{e^{\boldsymbol{\beta}_1 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

$$\Pr(Y_i = 2) = \frac{e^{\boldsymbol{\beta}_2 \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}}$$

$$\cdots\cdots$$

$$\Pr(Y_i = K-1) = \frac{e^{\boldsymbol{\beta}_{K-1} \cdot \mathbf{X}_i}}{1 + \sum_{k=1}^{K-1} e^{\boldsymbol{\beta}_k \cdot \mathbf{X}_i}} \quad \textbf{.....(Eq.1.1)}$$

1.4 Organization of Project Report:

This report is organized as follows:

Chapter 1 gives the Introduction to Background and Basics of Human Action identification, Literature Survey conducted, about Project to be undertaken, Problem Definition and Scope of the Project.

Chapter 2 gives an overview of Project Planning and Management. It states the detailed Functional Requirements, Non Functional Requirements, Deployment Requirements, External Interface Requirements and Other Requirements. This chapter also specifies the Project Process Model used to develop this Project, Cost and Effort estimates and Project Scheduling.

Chapter 3 gives the Analysis Study and Designing of the Project. It consists of IDEA Matrix, Mathematical Model, and all UML diagrams specifying how to build the required system.

Chapter 4 gives the details about implementation and coding. It consists of Operational Details, Major Classes and screenshots.

Chapter 5 gives the test cases identified to perform at final stage. Unit Testing, functional Testing and Manual Testing are performed on the system.

Chapter 6 consists of the Results obtained and Discussions

Chapter 7 gives the Conclusion.

Chapter 8 Mentions all the references used while building this system.

2: PROJECT PLANNING AND MANAGEMENT

2.1 Introduction:

This chapter covers the project planning and management details. It also covers System Requirement specifications. SRS is considered as the base for the effort estimations and project scheduling.

2.2 Detailed System Requirement Specification:

This section gives the detailed description of all types of requirements to be satisfied by the System to be developed.

2.2.1 System Overview:

The system to be developed is a python and machine Learning based Human Activity Recognition system.

The working of the systems is as follows:

The user can use the camera to identify its actions live.

The users can also upload previously recorded video to identify its actions.

2.2.2 Non Functional Requirements:

- **Performance Requirements**

The performance of the functions and every module must be well. The overall performance of the software will enable the users to work efficiently. Performance of encryption of data should be fast. Performance of the providing virtual environmentshould be fast Safety Requirement. The application is designed in modules where errors can be detected and corrected easily. This makes it easier to install and update new functionality if required.

- **Safety Requirement**

The application is designed in modules where errors can be detected and fixed easily. This makes it easier to install and update new functionality if required.

- **Software Quality Attributes:**

Our software has many quality attribute that are given below:-

  - Adaptability: This software is adaptable by all users.
  - Availability: This software is freely available to all users. The availability of the software is easy for everyone.

- Maintainability: After the deployment of the project if any error occurs then it can be easily maintained by the software developer.
- Reliability: The performance of the software is better which will increase the reliability of the Software.
- User Friendliness: Since, the software is a GUI application; the output generated is much user friendly in its behavior.
- Integrity: Integrity refers to the extent to which access to software or data by unauthorized persons can be controlled.
- Security: Users are authenticated using many security phases so reliable security is provided.
- Testability: The software will be tested considering all the test cases.

2.2.3 Software And Hardware Requirements:

- **HARDWARE:**

  RAM minimum required is 8 GB.Hard Disk: 40 GB

  Data Set of Video/Image is to be used hence minimum 40 GB Hard Disk memory is required.

  Processor: Intel i5 Processor

  Operating System: Windows 10

  Latest Operating System that supports all type of installation and development Environment

- **Software:**

  Python 3.7 64 bit – For preparing estimator object and training the model.

  Django 2.5 – Framework for deploying the model on web page.

  Pandas – For reading and refining the dataset and construct a dataframe.

  Mediapipe – For Human Pose Estimation And Keypoint Extraction

  Sklearn – For importing machine learning algorithms and metrics.

  Editor-HTML, javascript – For creating GUI.

  Browser – GUI platform.

2.3 Process Model:

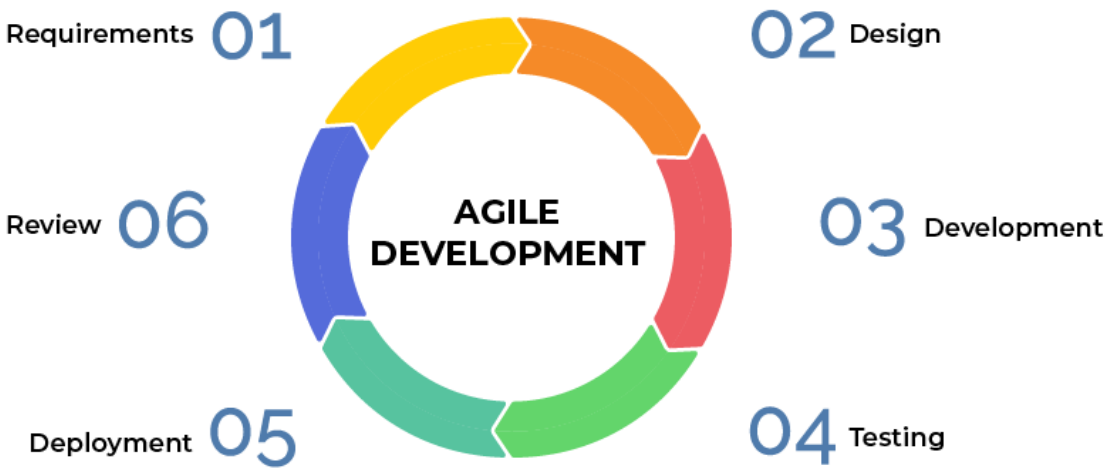


Figure 2.1 Agile Process Model

The process model used was **Agile Process model**. Software development approach was based on iterative development. The tasks were broken into smaller iterations; they did not have a long time planning. The project scope and requirements were laid down at the beginning of the development process. Focus was on quick responses to change and continuous development. An adaptive approach was used where there is no detailed planning and there is clarity on future tasks only in respect of what features need to be developed.

2.4 Cost and Effort Estimates:

Cost of the project will be the cost of hardware (mainly CPU) plus the cost of work that is put in. Basic COCOMO:

- Project class: We have determined our project is the characteristics of Semi-detached Mode as project is college level and requirements are rigid and less than rigid.
- Number of code Lines: We estimate our project will have 10000 Delivered Source instructions.

So, the Basic COCOMO model equations are as follows:

Effort Applied (E) = ab(KLOC) bb [man months]

Development Time (D) = cb (E) db [months]

People Required (P) = E/D [count]

Where:

KLOC (Kilo lines of code) is the estimated number of delivered lines (in thousands) of code for project.

E is the effort applied per person per month.

D is the development time in consecutive months.

The coefcients ab , bb , cb ,db are predetermined according to project class given in the following table:

Calculations: So, this project comes under the semi-detached mode.

Table 2.1 Cost Estimate Table

Software Project Class	a_b	b_b	c_b	d_b
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embeded	3.6	1.20	2.5	0.32

Efforts Applied (E) = 3.0*(10)(1.12) [man-months]

E = 39.54

Development Time (D) = 2.5* (39.54)(0.35) [months] D = 9.055

People Required (P) = 39.54/9.055 [count]

P = 4.36

2.5 Project Scheduling:

The System Implementation plan table shows the overall schedule of tasks compilation and time duration required for each task.

Table 2.2 Timeline Chart

			PHASE 1					PHASE 2			
ACTIVITY	**START**	**END**	**SEPT**	**OCT**	**NOV**	**DEC**	**JAN**	**FEB**	**MAR**	**APR**	**MAY**
Submission of project ideas.	**1/9/22**	**26/9/22**									
Approval of project idea.	**20/9/22**	**30/9/22**									
Initial Project Work Preparation	**1/10/22**	**25/11/22**									
First presentation about progress of project work(Review1 and Review2)	**2/12/22**	**30/12/22**									
Started Implementation											
Selection of technology Stack	5/1/22	19/1/22									
Pose Estimation Logic	15/1/22	10/3/22									
Front-end Development	25/2/22	6/5/22									
Classification model	3/4/22	25/4/22									
Link Frontend and Backend	10/4/22	15/5/22									
Manual Testing	25/3/22	3/4/22									
Automated Testing	3/5/22	16/5/22									
Review 3											
Presented partial implementation of project	25/3/22	28/3/22									
Review 4											
Presented complete implementation of project	2/5/22	6/5/22									
Submission of report for checking											
Make Revisions in Project report	13/5/22	20/5/22									
Final Submission of Report and Project											
Submitted Black-book Report copy	20/5/22										

3: ANALYSIS AND DESIGN

3.1 Introduction:

This chapter covers the analysis and design of the considered system.

System analysis is conducted for the purpose of studying a system or its parts in order to identify its objectives. It is a problem solving technique that improves the system and ensures that all the components of the system work efficiently to accomplish their purpose. Analysis specifies what the system should do.

Following are the diagrams that we've generated after to analyzing our system and we've designed our system accordingly.

3.2 System Architecture:

A system architecture diagram would be used to show the relationship between different components. Usually they are created for systems which include hardware and software and these are represented in the diagram to show the interaction between them.

USER: User can give live input to the system. It can also use pre-recorded video for Activity identification.

YSTEM: System uses a dataset which is then preprocessed. Using the dataset a csv file is created where details about the co-ordinates are written. Feature Extraction is done using mediapipe where all 33 body key points are extracted. Final action is recognized using Logistic Regression.

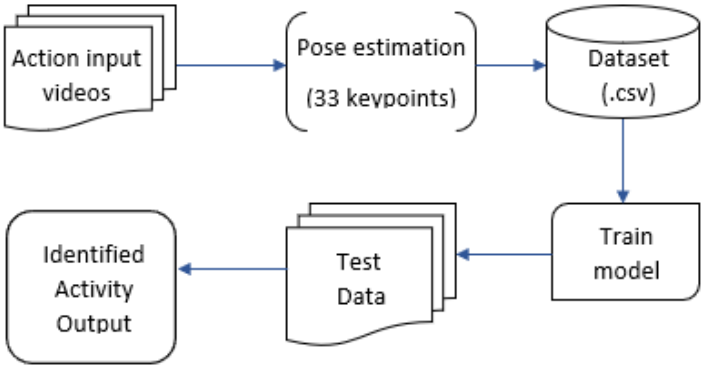


Figure 3.1 System Workflow

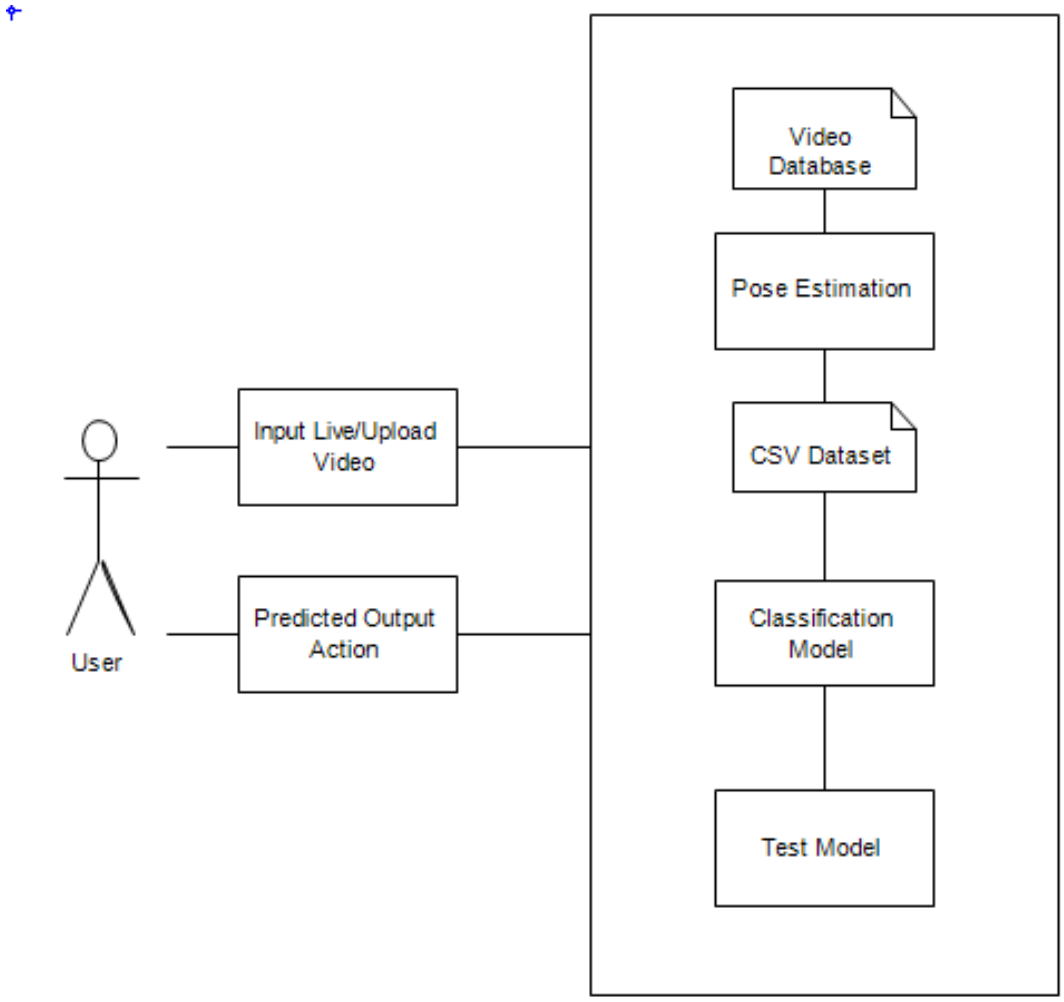


Figure 3.2 System Architecture

3.3 Use Case Diagram:

A use cases diagram at its simplest is a representation of a user's interaction with the system that shows the relationship between the user and the different use cases in which the user is involved.

USER: User gives live input to the system. It can also use pre-recorded video for Activity identification.

SYSTEM: Identifies and recognizes the actions.

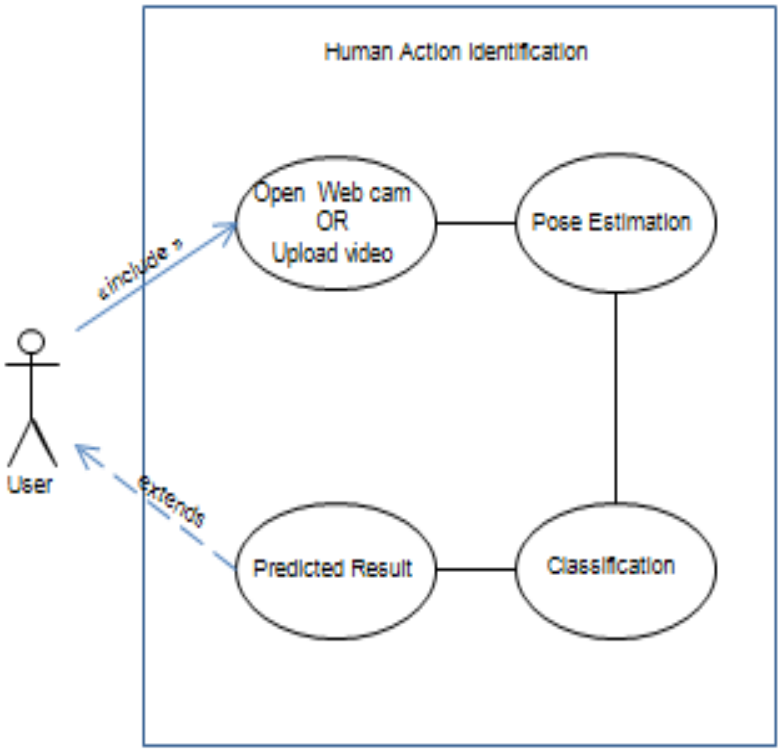


Figure 3.3 Use Case Diagram

3.4 Class Diagram:

Class diagram is a static diagram. It represents the static view of an application. We take live input from user or we can also use prerecorded video. csv file is created where details about coordinates are mentioned. Mediapipe is used to detect all 33 body key points. We have identified 6 actions namely Bend Down, Namaste, Hands on waist, Left Hand up, right Hand up, Standing. Logistic Regression is used for final action prediction. Action is identified

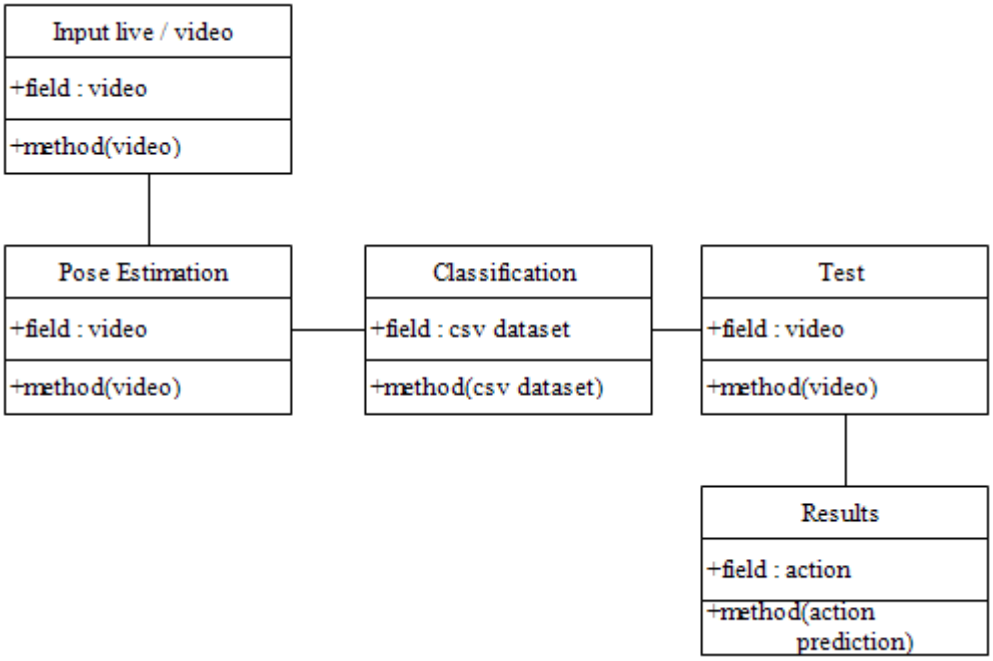


Figure 3.4 Class diagram

3.5 Activity Diagram:

Activity diagram is basically a flowchart to represent the flow from one activity to another activity. The activity can be described as an operation of the system.

The control flow is drawn from one operation to another. This flow can be sequential, branched, or concurrent. Activity diagrams deal with all type of flow control by using different elements such as fork, join, etc.

We take live input from user or we can also use prerecorded video. csv file is created where details about coordinates are mentioned. Mediapipe is used to detect all 33 body key points. We have identified 6 actions namely Bend Down, Namaste, Hands on waist, Left Hand up, right Hand up, Standing. Logistic Regression is used for final action prediction. Action is identified

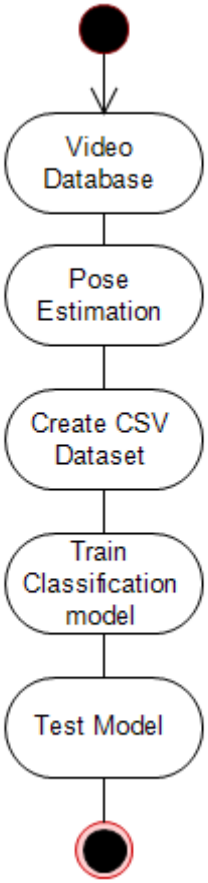


Figure 3.5 Activity Diagram

3.6 Sequence Diagram:

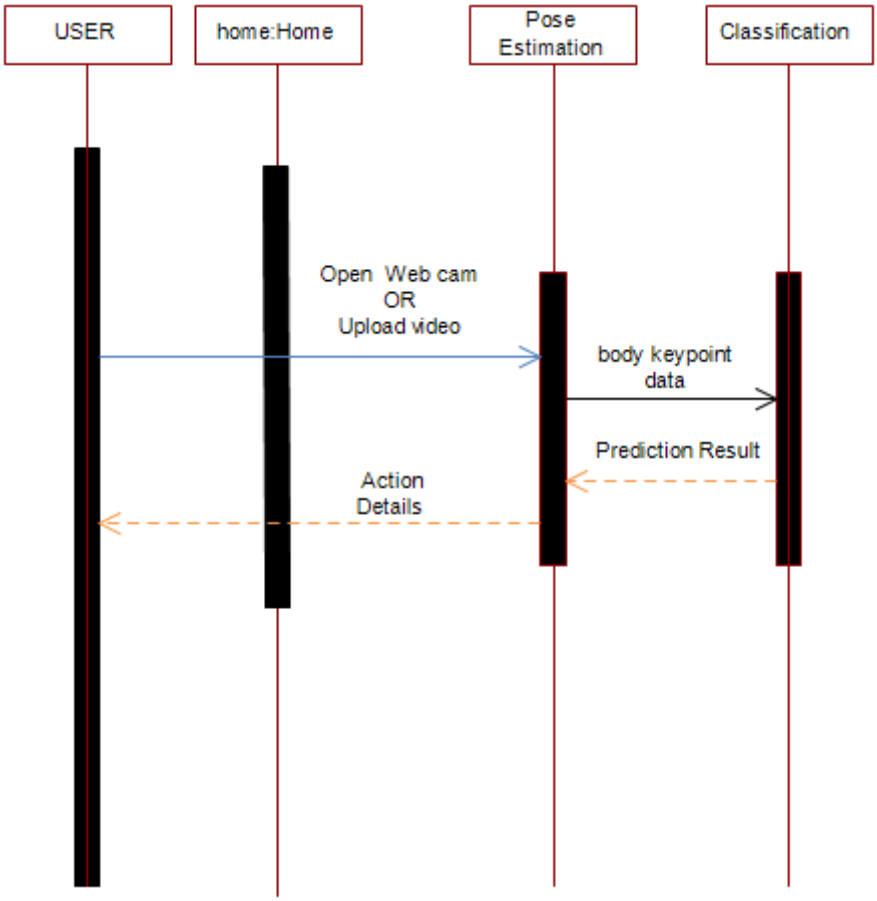


Figure 3.6 Sequence Diagram

3.7 Data Flow Diagram:

Figure 3.7 Data Flow(0) diagram

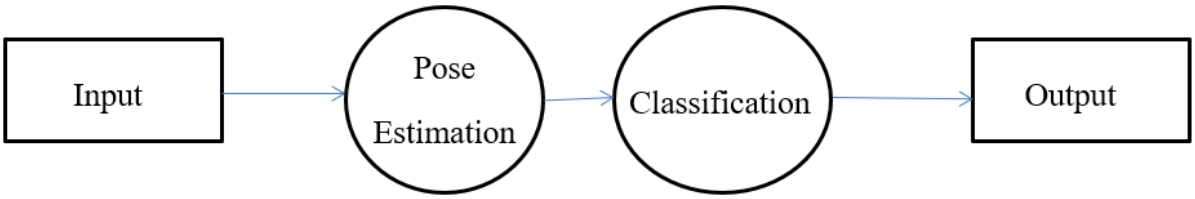


Figure 3.8 Data Flow(1) diagram

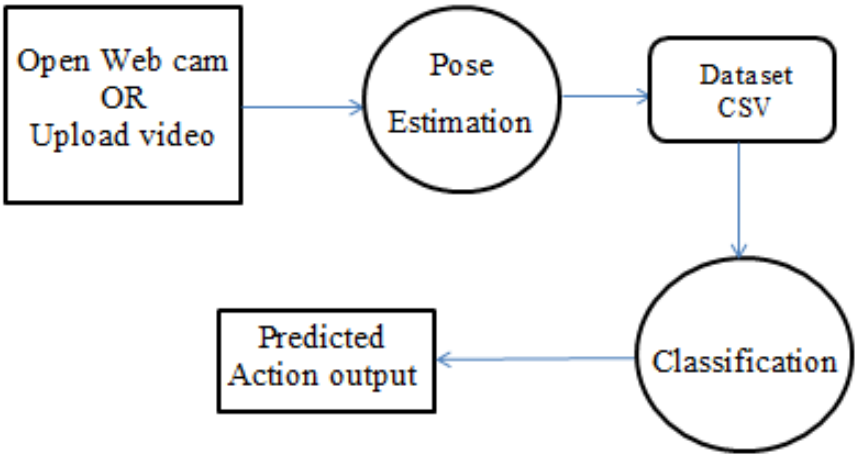


Figure 3.9 Data Flow(2) diagram

3.8 IDEA Matrix:

To develop the problem under consideration and justify feasibility using concepts of knowledge canvas and IDEA Matrix.

Table 3.1 I Elements

I	Use	Parameter Affected
Increase	Increase Accuracy of the training model by providing clear action video database.	Input video database
Iterate	Ideal value of Iterations in classification algorithm improves model quality	Classification algorithm

Table 3.2 D Elements

D	Use	Parameter Affected
Detect	Detect all actions accurately	Performance of system
Differentiate	Differentiate between different modules	User friendly UI

Table 3.3 E Elements

E	Use	Parameter Affected
Extract	Extract the body keypoints	Pose estimation
Exclude	Exclude the body keypoints that are not required for pose detection	CSV file creation

Table 3.4 A Elements

A	Use	Parameter Affected
Assurance	Assure that all body keypoints are identified correctly	Pose estimation
Associate	The system designed to Associate with video input	Action detection input

3.9 Mathematical Model:

Let S be the proposed system. S = { Input, Functions, Output, Success, Failure }

Where, Input ={I1}

Where,I1 = video database

Functions = {F1, F2, F3, F4}

Where,

F1 = Function to extract body keypoints using mediapipe

F2 = Function to create and load CSV file

F3 = Function to prepare ML Classification Model

F4 = Function to Test ML Model

Output = {O1, O2, O3 }

Where,

O1 = Body Keypoints extracted.

O2 = CSV file created.

O3 = ML Classification model created.

O4 = Tested ML model

Success ={S1, S2, S3}

Where,

S1 = Successfully identified and marked 33 body keypoints.

S2 = Successfully added the 33 body keypoint values to CSV file.

S3 = Successfully created the ML Classification model with high accuracy.

S4 = Successfully used the ML Classification on Test dataset and identified the action accurately

Failure = {Fl1, Fl2, Fl3}

Where,

Fl1 = Failed to extract body keypoint data.

Fl2 = Failed to create/load CSV file.

Fl3 = Failed to create ML Classification Model.

F14 = Failed to use ML model to identify action performed or Wrong action identified.

4: IMPLEMENTATION AND CODING

4.1 Introduction:

This chapter covers the role of various subsystems/modules/classes along with implementation details listing of the code for the major functionalities

4.2 Implementation:

- **Dataset Creation :**

```
import mediapipe as mp # Import mediapipe
import cv2 # Import opencv
import csv
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score # Accuracy metrics
import pickle
mp_drawing = mp.solutions.drawing_utils # Drawing helpers
mp_holistic = mp.solutions.holistic # Mediapipe Solutions

num_coords = len(results.pose_landmarks.landmark)
num_coords
landmarks = ['class']
for val in range(1, num_coords+1):
    landmarks += ['x{}'.format(val), 'y{}'.format(val), 'z{}'.format(val),
'v{}'.format(val)]

with open('coords.csv', mode='w', newline='') as f:
```

```
    csv_writer = csv.writer(f, delimiter=',', quotechar="",
quoting=csv.QUOTE_MINIMAL)
    csv_writer.writerow(landmarks)

class_name =
["Standing","Right_Hand_Up","Left_Hand_Up","Bend_Down","Namaste","Hands_
On_Waist"]
video_path =
["dataset/Standing.mp4","dataset/Right_Hand_Up.mp4","dataset/Left_Hand_Up.mp4
","dataset/Bend_Down.mp4","dataset/Namaste.mp4","dataset/Hands_On_Waist.mp4"
]

for i in range(0,len(class_name)):
    cap = cv2.VideoCapture(video_path[i])
    # Initiate holistic model
    with mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:

        while cap.isOpened():
            ret, frame = cap.read()
            if ret==True:
                # Recolor Feed
                image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                image.flags.writeable = False

                # Make Detections
                results = holistic.process(image)
                # print(results.face_landmarks)

                # face_landmarks, pose_landmarks, left_hand_landmarks,
right_hand_landmarks

                # Recolor image back to BGR for rendering
                image.flags.writeable = True
```

```
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # 4. Pose Detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                                      mp_drawing.DrawingSpec(color=(245,117,66),
thickness=2, circle_radius=4),
                                      mp_drawing.DrawingSpec(color=(245,66,230),
thickness=2, circle_radius=2)
                                      )
            # Export coordinates
            try:
               # Extract Pose landmarks
               pose = results.pose_landmarks.landmark
               pose_row = list(np.array([[landmark.x, landmark.y, landmark.z,
landmark.visibility] for landmark in pose]).flatten())



               # Append class name
               pose_row.insert(0, class_name[i])

               # Export to CSV
               with open('coords.csv', mode='a', newline='') as f:
                  csv_writer = csv.writer(f, delimiter=',', quotechar="",
quoting=csv.QUOTE_MINIMAL)
                  csv_writer.writerow(pose_row)

            except Exception as e:
               print("exeption : ",e)

            cv2.imshow('Raw Webcam Feed', image)
            cv2.waitKey(1)

         else:
```

```
            break

    cap.release()
    cv2.destroyAllWindows()
```

- **Model training and test**

```
df = pd.read_csv('coords.csv')

X = df.drop('class', axis=1) # features
y = df['class'] # target value

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=1234)



pipelines = {
    'lr':make_pipeline(StandardScaler(), LogisticRegression(max_iter=3000)),
    'rc':make_pipeline(StandardScaler(), RidgeClassifier(max_iter=3000)),
    'rf':make_pipeline(StandardScaler(), RandomForestClassifier()),
    'gb':make_pipeline(StandardScaler(), GradientBoostingClassifier()),
}

fit_models = {}
for algo, pipeline in pipelines.items():
    model = pipeline.fit(X_train, y_train)
    fit_models[algo] = model

for algo, model in fit_models.items():
    yhat = model.predict(X_test)
    print(algo, accuracy_score(y_test, yhat))

with open('body_language.pkl', 'wb') as f:
    pickle.dump(fit_models['rf'], f)
```

- **views.py**

```
from django.shortcuts import render,HttpResponse,redirect
from json import dumps
from HumanActionIdentification.settings import BASE_DIR
from .forms import Video_form
from .models import Video
from .load import inputCoordinates
from .apps import HaiConfig
import mediapipe as mp # Import mediapipe
import cv2 # Import opencv
import csv
import os
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler

from sklearn.linear_model import LogisticRegression, RidgeClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier
from sklearn.metrics import accuracy_score # Accuracy metrics
import pickle
# Create your views here.
def index(request):
    all_video=Video.objects.all()
    #print("inn")
    if request.method == "POST":
        #print("in if")
        form=Video_form(data=request.POST,files=request.FILES)

        if form.is_valid():
            print("in from if")
            form.save()
            return HttpResponse("<h1> Uploaded successfully </h1>")
```

```
            else:
                print(request.POST)
                print(request.FILES)
                print(form.non_field_errors)
                print(form.errors)
        else:
            #print("in else")
            form=Video_form()
        return render(request,'index.html',{"form":form,"all":all_video})

    def home(request):
        return render(request,'home.html')

    def videos(request):
        all_video=Video.objects.all()

        return render(request,'video.html',{"all":all_video})

    def predict(request):
        list1=[]
        mp_drawing = mp.solutions.drawing_utils # Drawing helpers
        mp_holistic = mp.solutions.holistic # Mediapipe Solutions
        cap = cv2.VideoCapture(0)
        #print(cv2.getBuildInformation())
        # Initiate holistic model
        with mp_holistic.Holistic(min_detection_confidence=0.5,
    min_tracking_confidence=0.5) as holistic:
            #print("In cap is opened")
            while cap.isOpened():
                ret, frame = cap.read()
                #print("In cap is opened")
                # Recolor Feed
                image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                image.flags.writeable = False
```

```
        # Make Detections
        results = holistic.process(image)
        # print(results.face_landmarks)

        # face_landmarks, pose_landmarks, left_hand_landmarks, right_hand_landmarks

        # Recolor image back to BGR for rendering
        image.flags.writeable = True
        image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

        # 1. Pose Detections
        mp_drawing.draw_landmarks(image, results.pose_landmarks, mp_holistic.POSE_CONNECTIONS,
                                  mp_drawing.DrawingSpec(color=(245,117,66), thickness=2, circle_radius=4),
                                  mp_drawing.DrawingSpec(color=(245,66,230), thickness=2, circle_radius=2)
                                  )
        # Export coordinates
        try:
            # Extract Pose landmarks
            pose = results.pose_landmarks.landmark
            pose_row = list(np.array([[landmark.x, landmark.y, landmark.z, landmark.visibility] for landmark in pose]).flatten())

            row = pose_row

    #         # Append class name
    #         row.insert(0, class_name)

    #         # Export to CSV
    #         with open('coords.csv', mode='a', newline='') as f:
```

```
    #                 csv_writer = csv.writer(f, delimiter=',', quotechar="",
quoting=csv.QUOTE_MINIMAL)
    #             csv_writer.writerow(row)

            # Make Detections
            #print('hello')
            X = pd.DataFrame([row])
            body_language_class = HaiConfig.model.predict(X)[0]
            body_language_prob = HaiConfig.model.predict_proba(X)[0]
            #print(body_language_class, body_language_prob)
            list1.append(inputCoordinates(body_language_class,body_language_prob))
            #print('after')
            # Grab ear coords
            coords = tuple(np.multiply(
                        np.array(

(results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,

results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y))
                    , [640,480]).astype(int))

            cv2.rectangle(image,
                    (coords[0], coords[1]+5),
                    (coords[0]+len(body_language_class)*20, coords[1]-30),
                    (245, 117, 16), -1)
            cv2.putText(image, body_language_class, coords,
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,
cv2.LINE_AA)

            # Get status box
            cv2.rectangle(image, (0,0), (250, 60), (245, 117, 16), -1)

            # Display Class
            cv2.putText(image, 'CLASS'
```

```
                    , (95,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
            cv2.putText(image, body_language_class.split(' ')[0]
                    , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,
cv2.LINE_AA)

            # Display Probability
            cv2.putText(image, 'PROB'
                    , (15,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
            cv2.putText(image,
str(round(body_language_prob[np.argmax(body_language_prob)],2))
                    , (10,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,
cv2.LINE_AA)



        except Exception as e:
            #print("exeption : ",e)
            pass

        cv2.imshow('Raw Webcam Feed', image)

        if cv2.waitKey(10) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
    return render(request,'predResult.html',{'list':list1})

def predictUploaded(request):
    list1=[]
    #lastvideo= Video.objects.last()
```

```
    val1=str(request.POST["sel"])
    #print("sel : ",val1)
    p='\\Users\\admin\\Desktop\\BEproj\\HumanActionIdentification'+val1

path=os.path.join(os.path.join(BASE_DIR,p))#os.path.dirname(__file__),'Input_Chec
k.mp4')
    #print(path)

    #print(lastvideo.video.url)
    #print(os.path.join(BASE_DIR,lastvideo.video.url))
    mp_drawing = mp.solutions.drawing_utils # Drawing helpers
    mp_holistic = mp.solutions.holistic # Mediapipe Solutions
    cap = cv2.VideoCapture(path)#lastvideo.video.url)
    # Initiate holistic model
    with                    mp_holistic.Holistic(min_detection_confidence=0.5,
min_tracking_confidence=0.5) as holistic:
        #print("in with")
        while cap.isOpened():
            #print("in isopend")
            ret, frame = cap.read()
            if ret==True:
                # Recolor Feed
                image = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                image.flags.writeable = False

                # Make Detections
                results = holistic.process(image)
                # print(results.face_landmarks)

                #       face_landmarks,       pose_landmarks,       left_hand_landmarks,
right_hand_landmarks

                # Recolor image back to BGR for rendering
                image.flags.writeable = True
```

```
            image = cv2.cvtColor(image, cv2.COLOR_RGB2BGR)

            # 1. Pose Detections
            mp_drawing.draw_landmarks(image, results.pose_landmarks,
mp_holistic.POSE_CONNECTIONS,
                                      mp_drawing.DrawingSpec(color=(245,117,66), thickness=2,
circle_radius=4),
                                      mp_drawing.DrawingSpec(color=(245,66,230), thickness=2,
circle_radius=2)
                                      )
            # Export coordinates
            try:
               # Extract Pose landmarks
               pose = results.pose_landmarks.landmark
               pose_row = list(np.array([[landmark.x, landmark.y, landmark.z,
landmark.visibility] for landmark in pose]).flatten())

               row = pose_row

      #          # Append class name
      #          row.insert(0, class_name)

      #          # Export to CSV
      #          with open('coords.csv', mode='a', newline='') as f:
      #                       csv_writer = csv.writer(f, delimiter=',', quotechar='',
quoting=csv.QUOTE_MINIMAL)
      #              csv_writer.writerow(row)

               # Make Detections
               #print('hello')
               X = pd.DataFrame([row])
               body_language_class = HaiConfig.model.predict(X)[0]
               body_language_prob = HaiConfig.model.predict_proba(X)[0]
               #print(body_language_class, body_language_prob)
```

```
list1.append(inputCoordinates(body_language_class,body_language_prob))
                #print('after')
                # Grab ear coords
                coords = tuple(np.multiply(
                        np.array(

(results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].x,

results.pose_landmarks.landmark[mp_holistic.PoseLandmark.LEFT_EAR].y))
                        , [640,480]).astype(int))

                cv2.rectangle(image,
                        (coords[0], coords[1]+5),
                        (coords[0]+len(body_language_class)*20, coords[1]-30),
                        (245, 117, 16), -1)
                cv2.putText(image, body_language_class, coords,
                        cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2,
cv2.LINE_AA)

                # Get status box
                cv2.rectangle(image, (0,0), (250, 60), (245, 117, 16), -1)

                # Display Class
                cv2.putText(image, 'CLASS'
                        , (95,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
                cv2.putText(image, body_language_class.split(' ')[0]
                        , (90,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255),
2, cv2.LINE_AA)

                # Display Probability
                cv2.putText(image, 'PROB'
```

```
                        , (15,12), cv2.FONT_HERSHEY_SIMPLEX, 0.5, (0, 0, 0), 1,
cv2.LINE_AA)
                cv2.putText(image,
str(round(body_language_prob[np.argmax(body_language_prob)],2))
                        , (10,40), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255),
2, cv2.LINE_AA)



            except Exception as e:
                #print("exeption : ",e)
                pass

            cv2.imshow('Raw Webcam Feed', image)
            if cv2.waitKey(10) & 0xFF == ord('q'):
                break

        else:
            break

    cap.release()
    cv2.destroyAllWindows()
    return render(request,'predResult.html',{'list':list1})

def help(request):
    return render(request,'help.html')

def delete(request):
    Video.objects.all().delete()
    return render(request,'video.html')
```

- **Functional testing :**

test_project_blank_page.py

```
from selenium import webdriver
from selenium.webdriver.common.action_chains import ActionChains
from django.contrib.staticfiles.testing import StaticLiveServerTestCase
from django.urls import reverse
import time

class TestProjectBlankPage(StaticLiveServerTestCase):

    def setUp(self):
        self.browser=webdriver.Chrome('functional_tests/chromedriver.exe')

    def tearDown(self):
        self.browser.close()

    def test_dashboard_is_displayed(self):
        self.browser.get(self.live_server_url+reverse('home'))
        time.sleep(10)

    def test_dashboard_button_redirects_to_home_page(self):
        self.browser.get(self.live_server_url+reverse('home'))

        home_url=self.live_server_url + reverse('home')
        self.browser.find_element_by_class_name('dashboard').click()
        self.assertEquals(
            self.browser.current_url,
            home_url
        )

    def test_index_button_redirects_to_index_page(self):
        self.browser.get(self.live_server_url+ reverse('home'))
```

```
        index_url=self.live_server_url + reverse('index')
        button=self.browser.find_element_by_class_name('index').click()
        self.assertEquals(
            self.browser.current_url,
            index_url
        )

    def test_help_button_redirects_to_help_page(self):
        self.browser.get(self.live_server_url+ reverse('home'))

        help_url=self.live_server_url + reverse('help')
        self.browser.find_element_by_class_name('help').click()
        self.assertEquals(
            self.browser.current_url,
            help_url
        )
```

- **Unit testing:**

test_urls.py

```
from django.test import SimpleTestCase
from django.urls import reverse,resolve
from HAI.views import index,home,delete,predictUploaded,predict,videos,help

class TestUrls(SimpleTestCase):

    def test_index_url_resolves(self):
        #assert 1==2
        url=reverse('index')
        self.assertEquals(resolve(url).func,index)
        print('Test : test_index_url_resolves')
```

```
    def test_home_url_resolves(self):
        url=reverse('home')
        self.assertEquals(resolve(url).func,home)
        print('Test : test_home_url_resolves')

    def test_predict_url_resolves(self):
        url=reverse('predict')
        self.assertEquals(resolve(url).func,predict)
        print('Test : test_predict_url_resolves')

    def test_videos_url_resolves(self):
        url=reverse('videos')
        self.assertEquals(resolve(url).func,videos)
        print('Test : test_videos_url_resolves')

    def test_predictUploaded_url_resolves(self):
        url=reverse('predictUploaded')
        self.assertEquals(resolve(url).func,predictUploaded)
        print('Test : test_predictUploaded_url_resolves')


    def test_help_url_resolves(self):
        url=reverse('help')
        self.assertEquals(resolve(url).func,help)
        print('Test : test_help_url_resolves')

    def test_delete_url_resolves(self):
        url=reverse('delete')
        self.assertEquals(resolve(url).func,delete)
        print('Test : test_delete_url_resolves')
```

test_views.py

```
from django.test import TestCase,Client
from django.urls import reverse
import json

class TestViews(TestCase):

    def setUp(self):
        self.client=Client()
        self.index_url=reverse('index')
        self.home_url=reverse('home')
        self.predictUploaded_url=reverse('predictUploaded')
        self.predict_url=reverse('predict')
        self.help_url=reverse('help')
        self.delete_url=reverse('delete')
        self.videos_url=reverse('videos')

    def test_index_GET(self):
        responce=self.client.get(self.index_url)

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'index.html')
        print('Test : test_index_GET')

    def test_home_GET(self):
        responce=self.client.get(self.home_url)

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'home.html')
        print('Test : test_home_GET')

    def test_delete(self):
        responce=self.client.post(self.delete_url)
```

```
        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'video.html')
        print('Test : test_delete')

    def test_video(self):
        responce=self.client.post(self.videos_url)

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'video.html')
        print('Test : test_video')

    def test_predict(self):
        responce=self.client.post(self.predict_url)

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'predResult.html')
        print('Test : test_predict')

    def test_help_GET(self):
        responce=self.client.get(self.help_url)

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'help.html')
        print('Test : test_help_GET')

    def test_predictUploaded_POST(self):
        responce=self.client.post(self.predictUploaded_url,{
            'sel':'media\HAI\Input_Check.mp4'
        })

        self.assertEquals(responce.status_code,200)
        self.assertTemplateUsed(responce,'predResult.html')
        print('Test : test_predictUploaded_POST')
```

Flowchart:

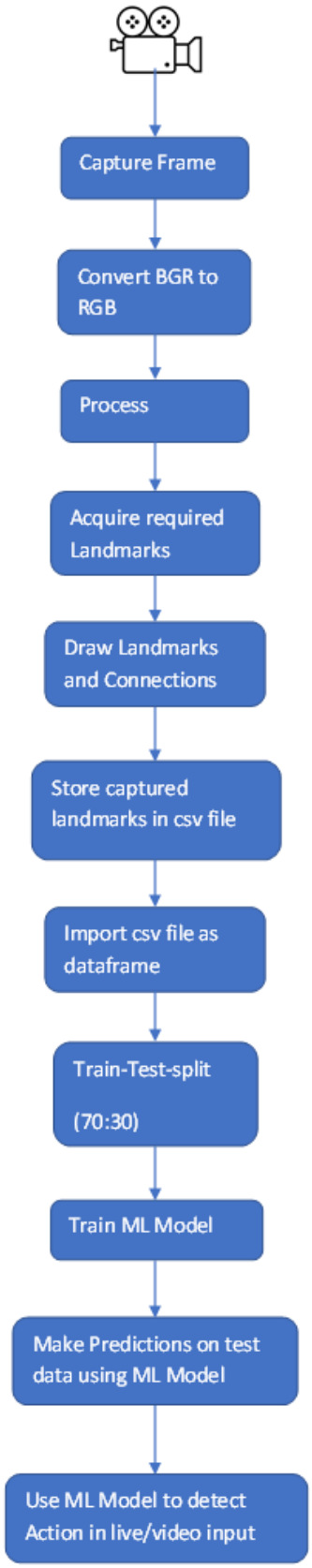


Figure 4.1 Flowchart of system

5: TESTING

5.1 Introduction:

Software Testing is a method to check whether the actual software product matches expected requirements and to ensure that software product is Defect free. It involves execution of software/system components using manual or automated tools to evaluate one or more properties of interest. The purpose of software testing is to identify errors, gaps or missing requirements in contrast to actual requirements

Testing is a method of assessing the functionality of a software program. This chapter covers the testing approach used and the test cases.

We've performed unit testing, functional testing, and manual testing to test our system.

5.2 Unit Testing:

Unit Testing is a level of software testing where individual units/components of software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output.

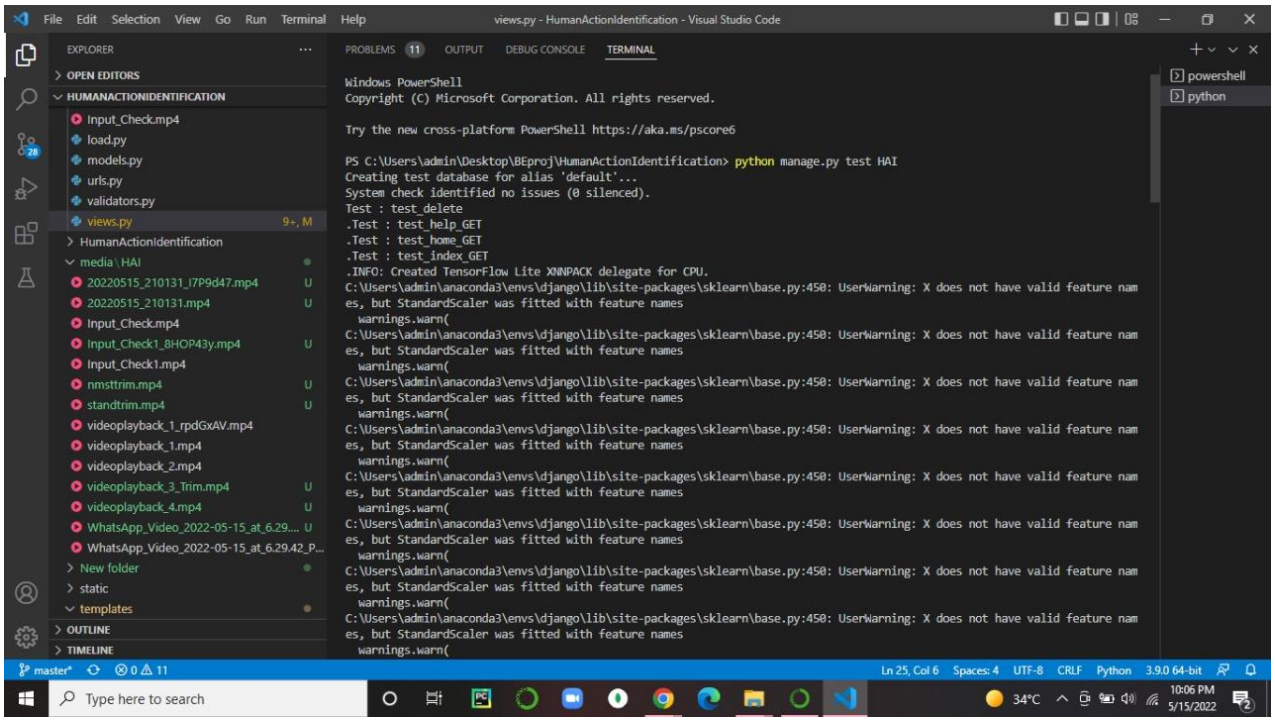


Figure 5.1 Unit Testing 1

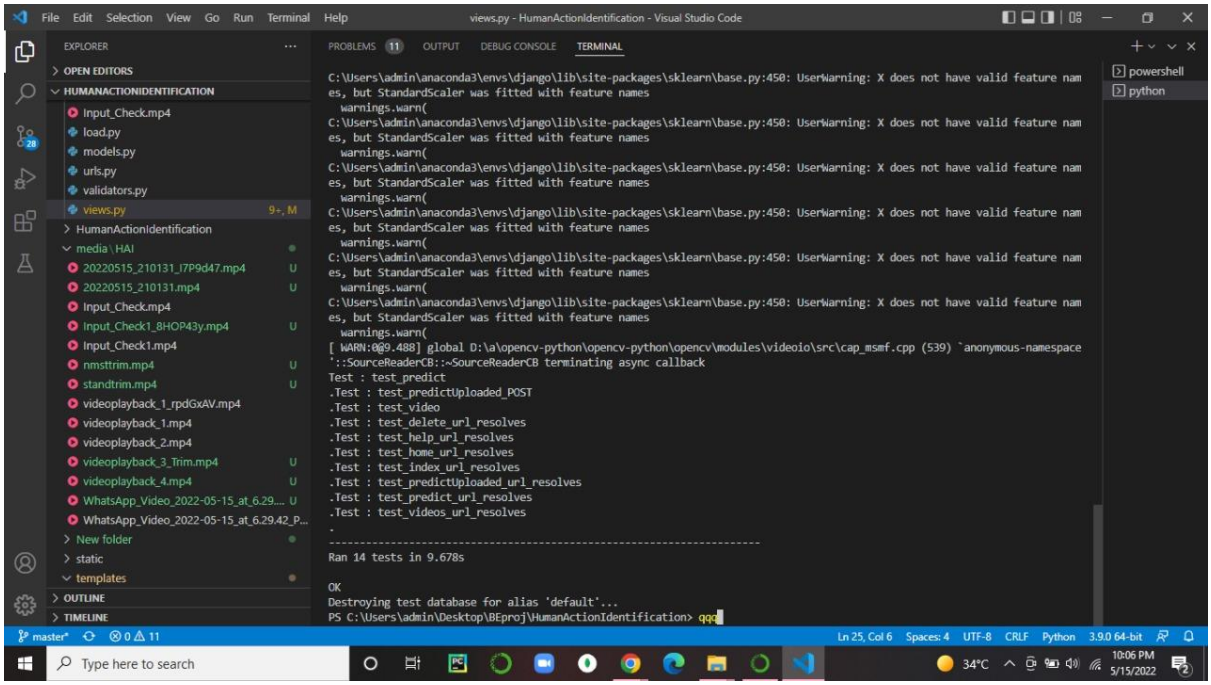


Figure 5.2 Unit Testing 2

5.3 Functional Testing:

Functional testing is a type of testing that seeks to establish whether each application feature works as per the software requirements. Each function is compared to the corresponding requirement to ascertain whether its output is consistent with the end user's expectations. The testing is done by providing sample inputs, capturing resulting outputs, and verifying that actual outputs are the same as expected outputs.

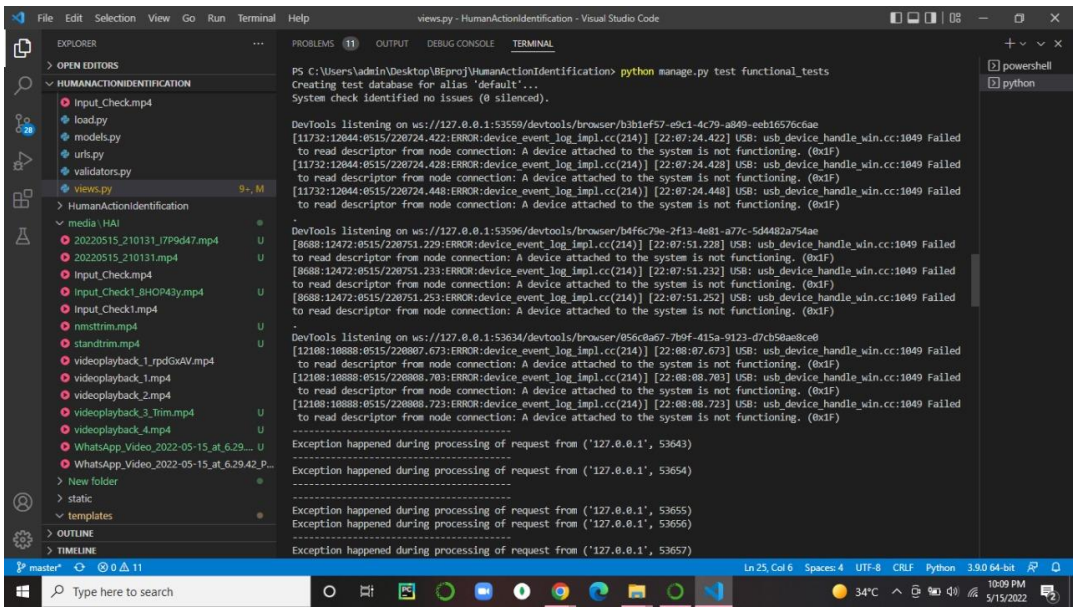


Figure 5.3 Functional Testing 1

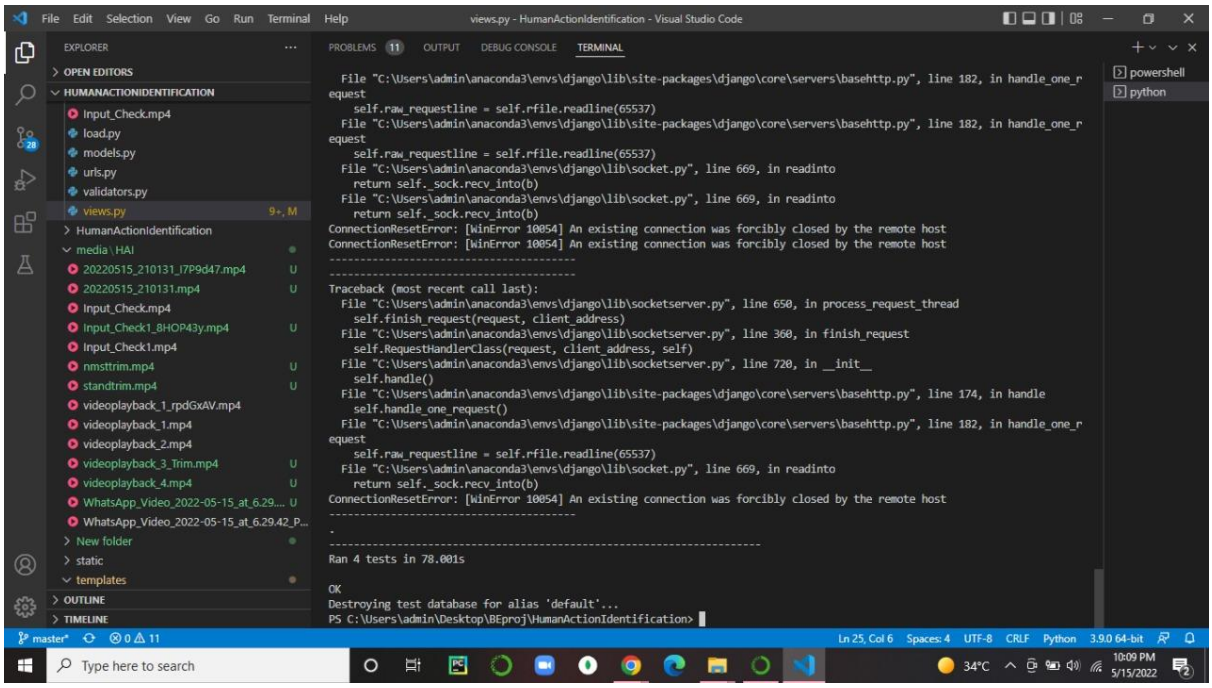


Figure 5.4 Functional Testing 2

5.4 Manual Testing:

Manual testing is a software testing process in which test cases are executed manually without using any automated tool. All test cases executed by the tester manually according to the end user's perspective. It ensures whether the application is working, as mentioned in the requirement document or not. Test cases are planned and implemented to complete almost 100 percent of the software application. Test case reports are also generated manually.

Table 5.1 Manual Testing Test Cases

SR NO	Description	Expected Output	Actual Output	Result
1	Dashboard should be displayed	Dashboard displayed	Dashboard displayed	Pass
2	After clicking on help, Help tab should be displayed	Help displayed	Help displayed	Pass

3	After clicking on Action Identification, Action Identification page should be displayed	Action Identification page displayed	Action Identification page displayed	Pass
4	After clicking on live input button, webcam should start to record the input	Webcam starts to record the input	Webcam starts to record the input	Pass
5	If the person is standing, system should identify action as standing	Action identified as standing	Action identified as standing	Pass
6	If the person has its hands on waist, system should identify action as hands on waist	Action identified as hands on waist	Action identified as hands on waist	Pass
7	If the person is doing namaste, system should identify action as namaste	Action identified as Namaste	Action identified as namaste	Pass
8	If the persons right hand is up, system should identify action right hand up	Action identified as right hand up	Action identified as right hand up	Pass
9	If the persons left hand is up, system should identify action as left hand up	Action identified as left hand up	Action identified as left hand up	Pass
10	If the person is bending down, system should identify action as bending down	Action identified as bending down	Action identified as bending down	Pass

11	User should be able to select a video to upload	User can select a video to upload	User can select a video to upload	Pass
12	Video should be uploaded successfully	Video uploaded successfully	Video uploaded successfully	Pass
13	Users should be able to write captions for each video	Users are able to write captions for each video	Users are able to write captions for each video	Pass
14	All uploaded videos along with their captions should be visible	All uploaded videos along with their captions are visible	All uploaded videos along with their captions are visible	Pass
15	User should be able to select one of the uploaded videos for action identification	User are able to select one of the uploaded videos for action identification	User are able to select one of the uploaded videos for action identification	Pass
16	Actions in video should be identified	Actions in video are identified	Actions in video are identified	Pass
17	If no person is detected in the video, system should print Action Unidentified message	System prints Action Unidentified message	System prints Action Unidentified message	Pass
18	Video uploaded should be less than 50 MB	Video is uploaded successfully if video size is less than 50 MB.	Video is uploaded successfully if video size is less than 50 MB.	Pass

6: RESULTS AND DISCUSSIONS

6.1 Result:

We have used Logistic regression, ridge classifier, random forest classifier, gradient boosting classifier and got 100% accuracy for Logistic regression, random forest classifier, gradient boosting classifier and 99% for ridge classifier. We used logistic regression as our final model for prediction.

Table 6.4 Evaluation of Algorithms

Algorithm	Precision	Recall	F1-Score	Support
Logistic regression	1.0	1.0	1.0	651
Ridge classifier	0.99	0.99	0.99	651
Random forest classifier	1.0	1.0	1.0	651
Gradient boosting classifier	1.0	1.0	1.0	651

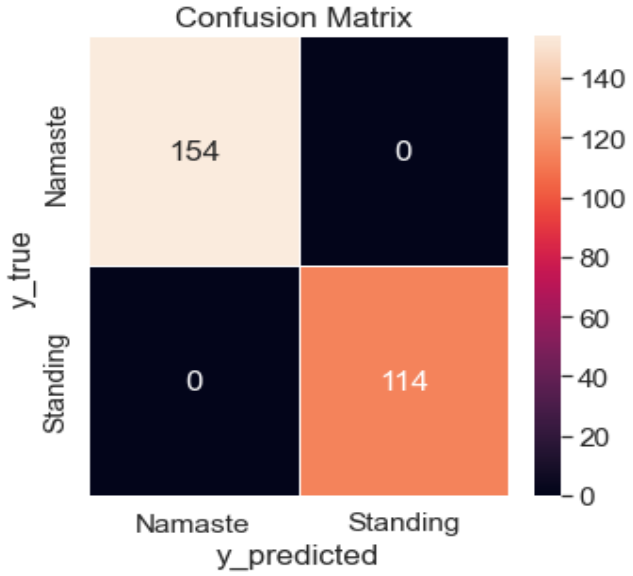


Figure 6.1 Confusion Matrix 1

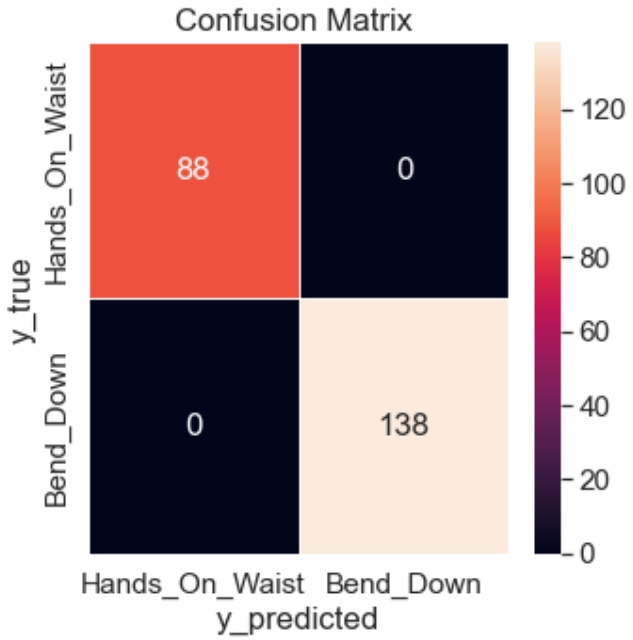


Figure 6.2 Confusion Matrix 2

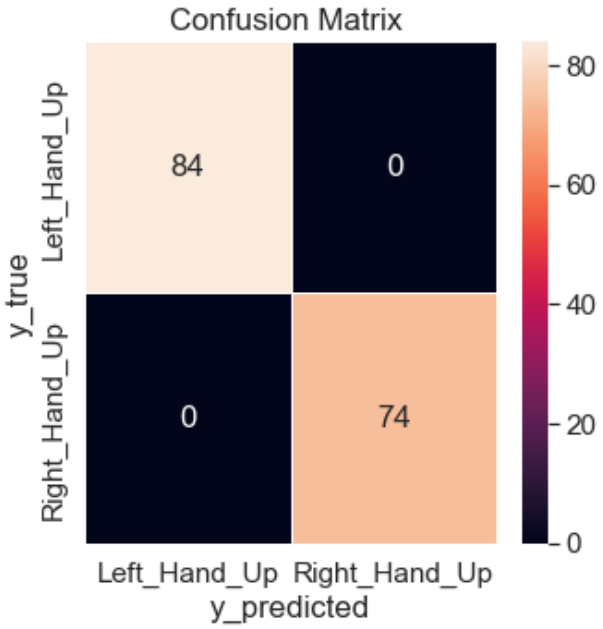


Figure 6.3 Confusion Matrix 3

6.2 Main GUI Snapshots:

Dashboard:

Basic Information regarding project is displayed

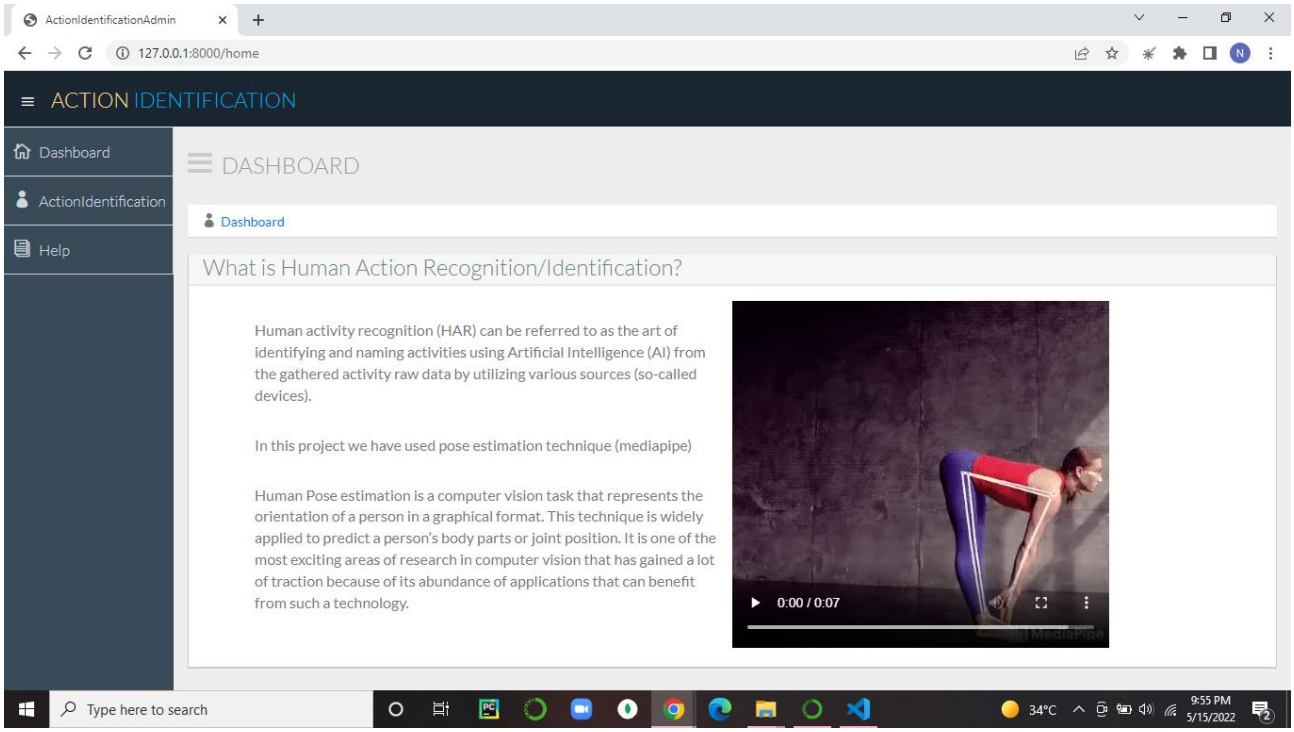


Figure 6.4 Dashboard

Action Identification model:

We have two options:

Take Live Input

Upload Video

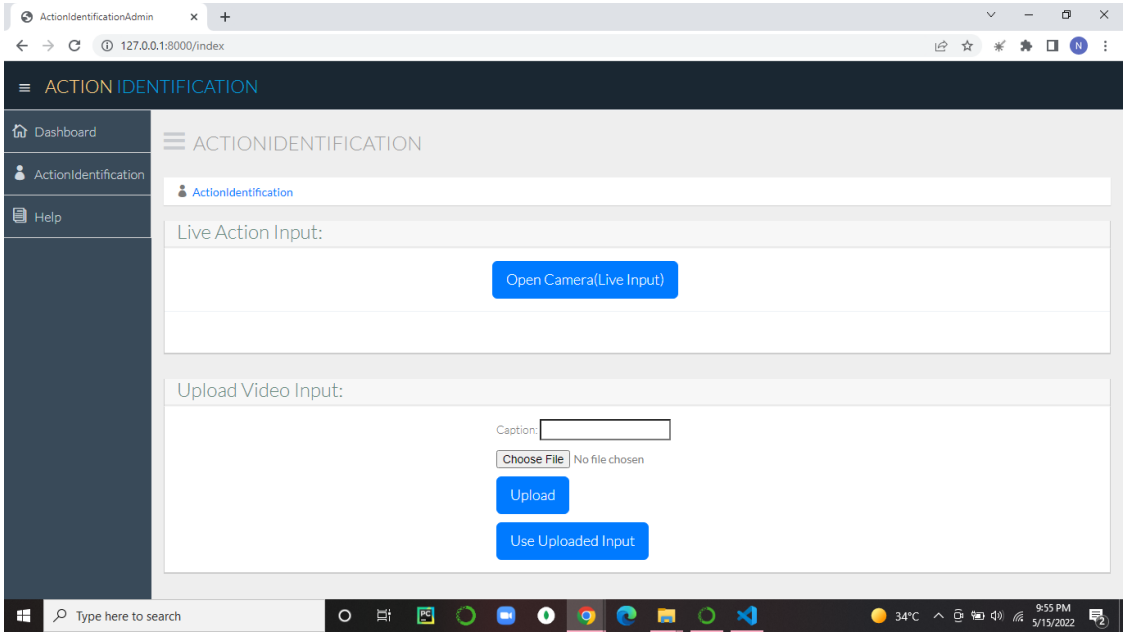


Figure 6.5 Action Identification model

Help Tab:

Directions to use the software are mentioned here.

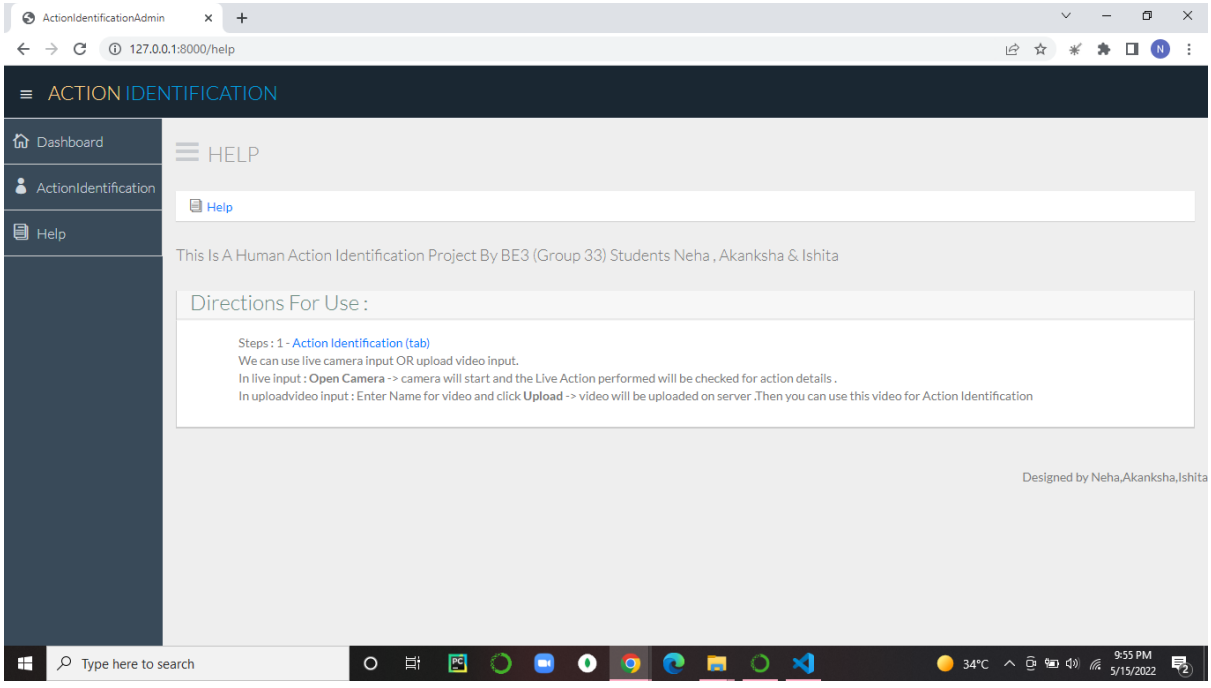


Figure 6.6 Help Tab

Upload Videos:

Select video to upload for activity identification.

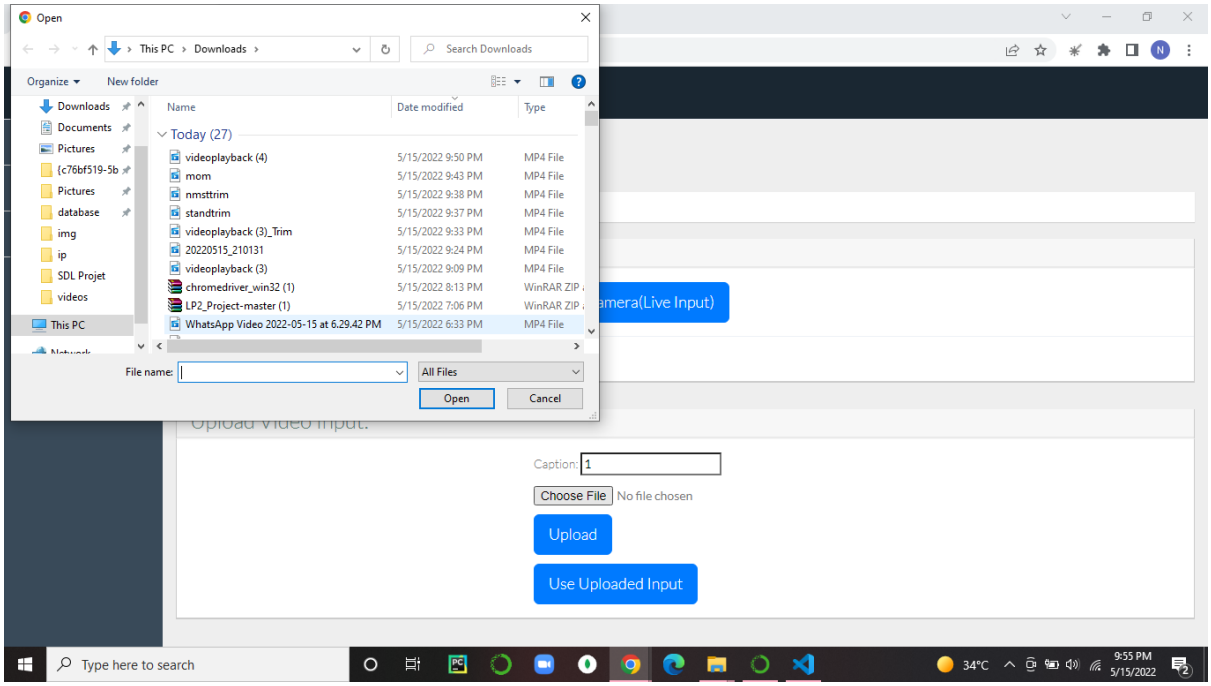


Figure 6.7 Upload Videos

Video is uploaded successfully

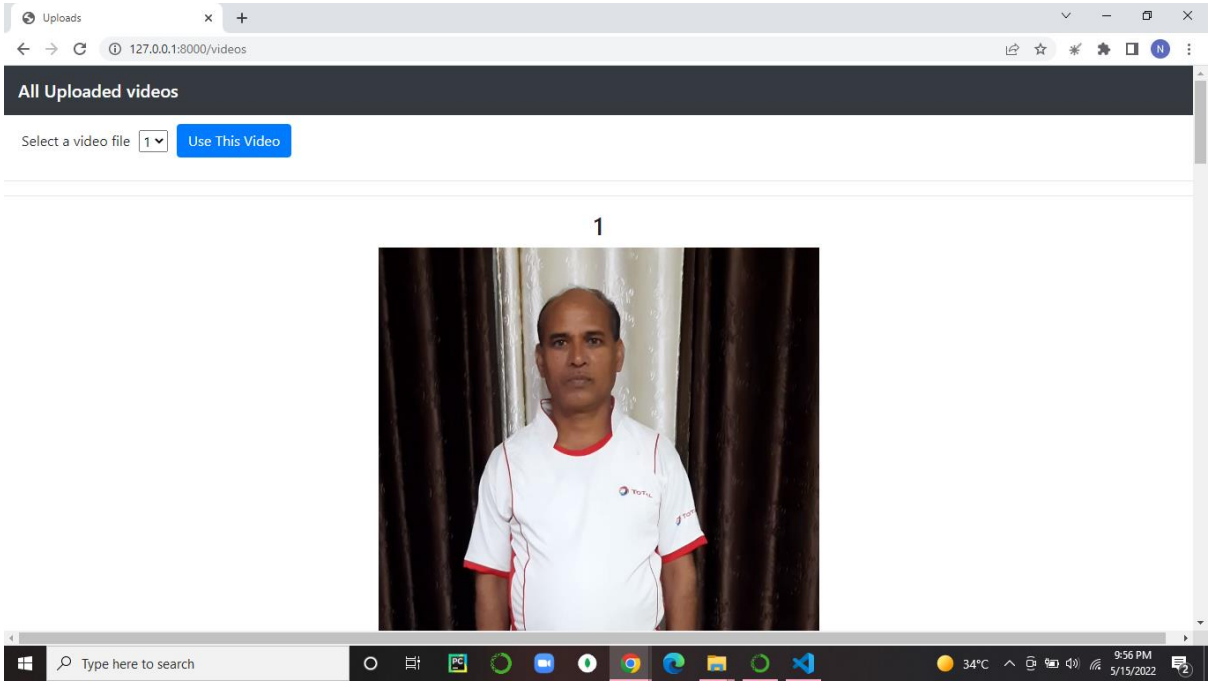


Figure 6.8 View all uploaded videos

Action detected:

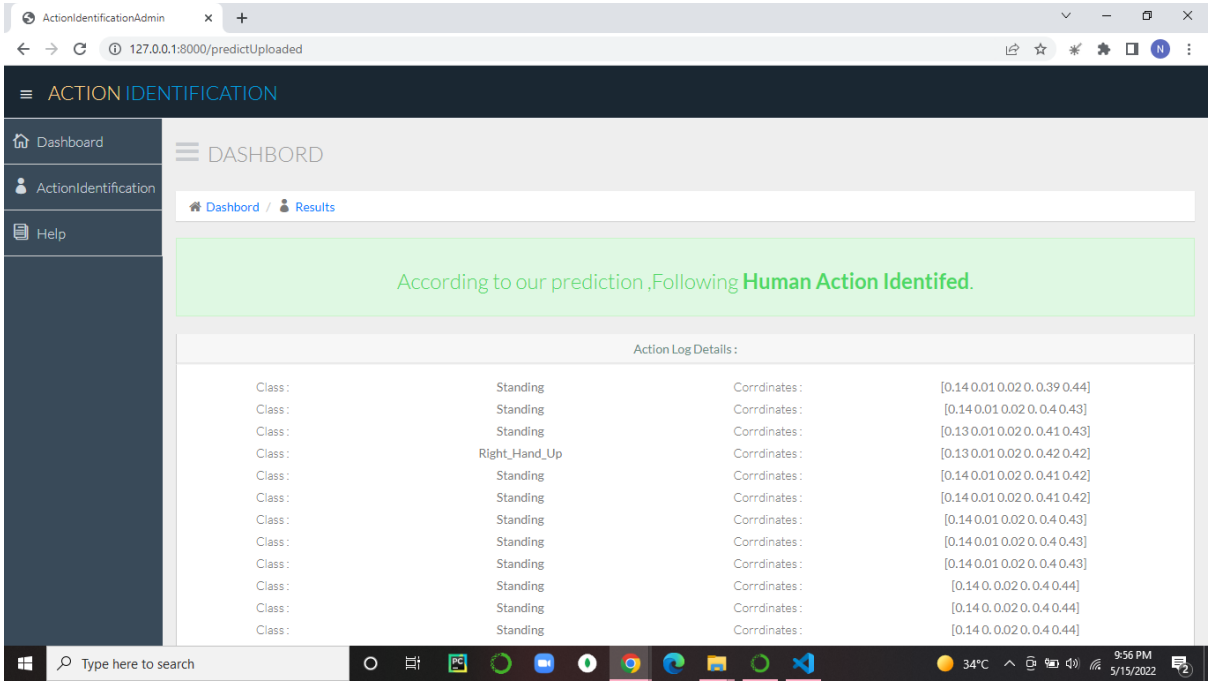


Figure 6.9 Action Detected

Action Detected:

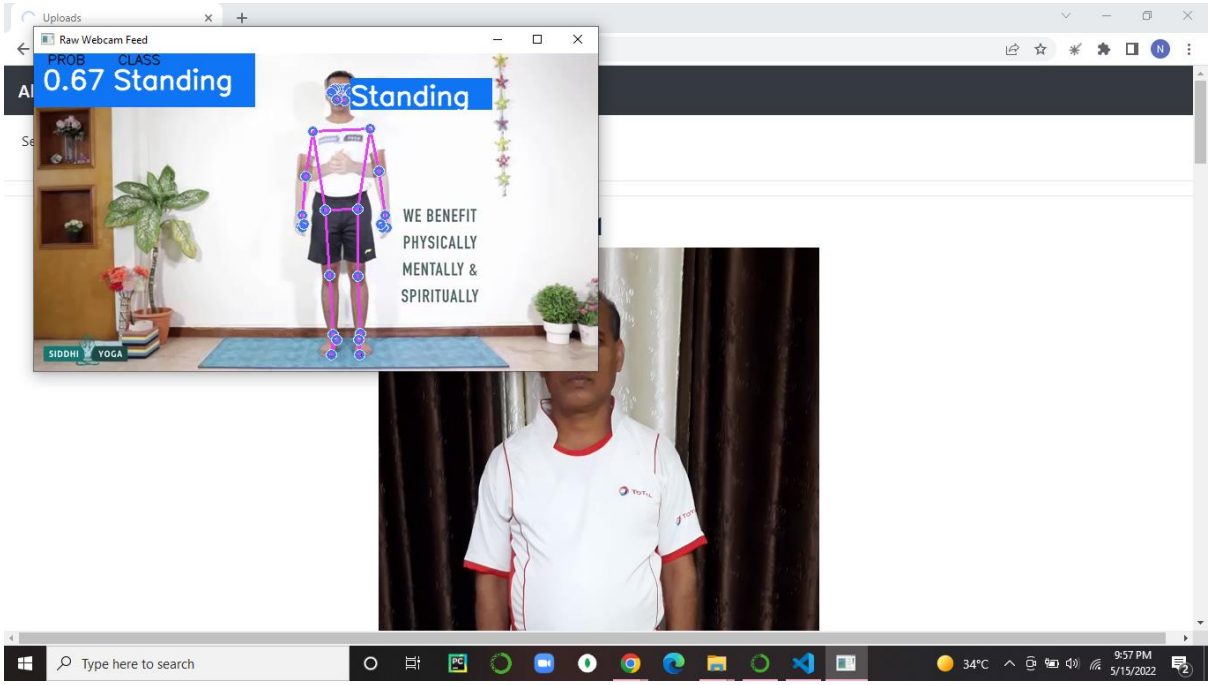


Figure 6.10 Action Detected Standing

No action identified if no human is detected:

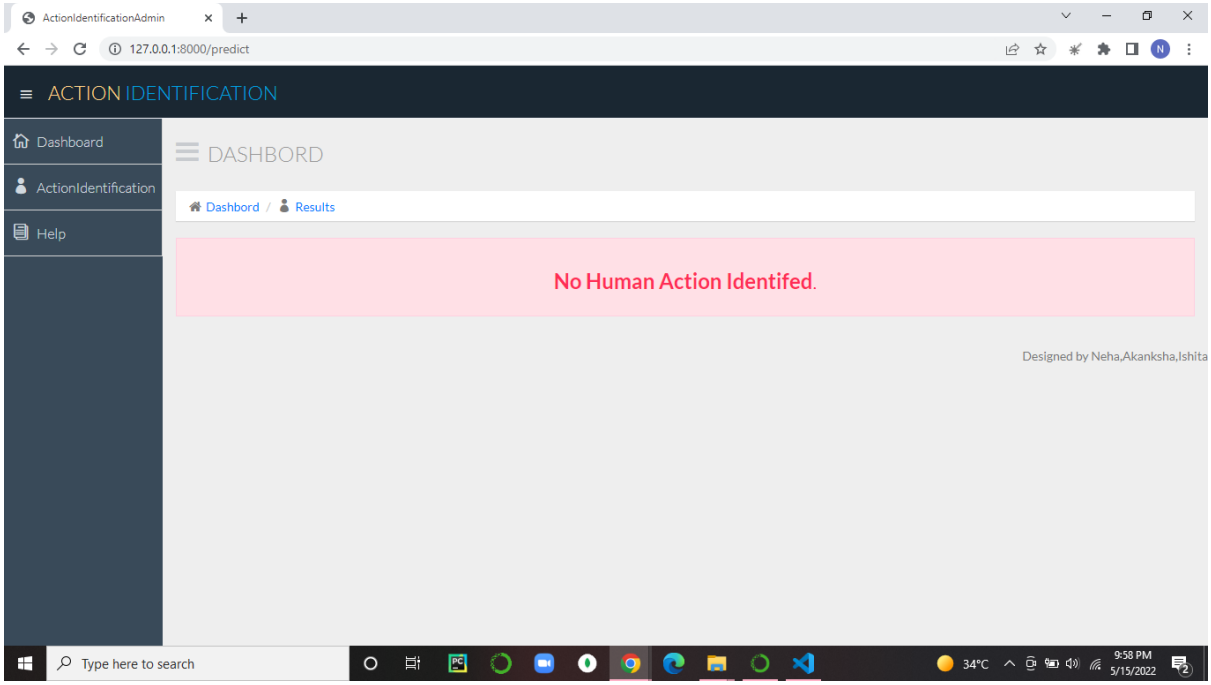


Figure 6.11 No Action Detected

Action detected using live input (Webcam):

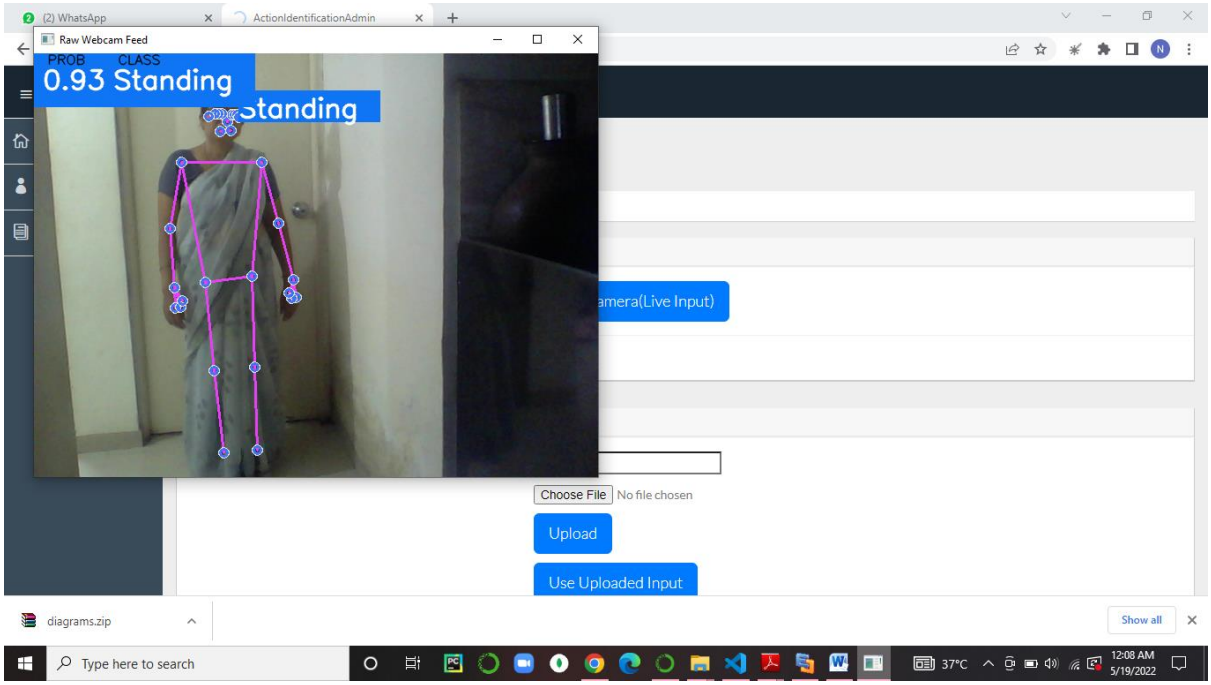


Figure 6.12 Live video Input (Standing)

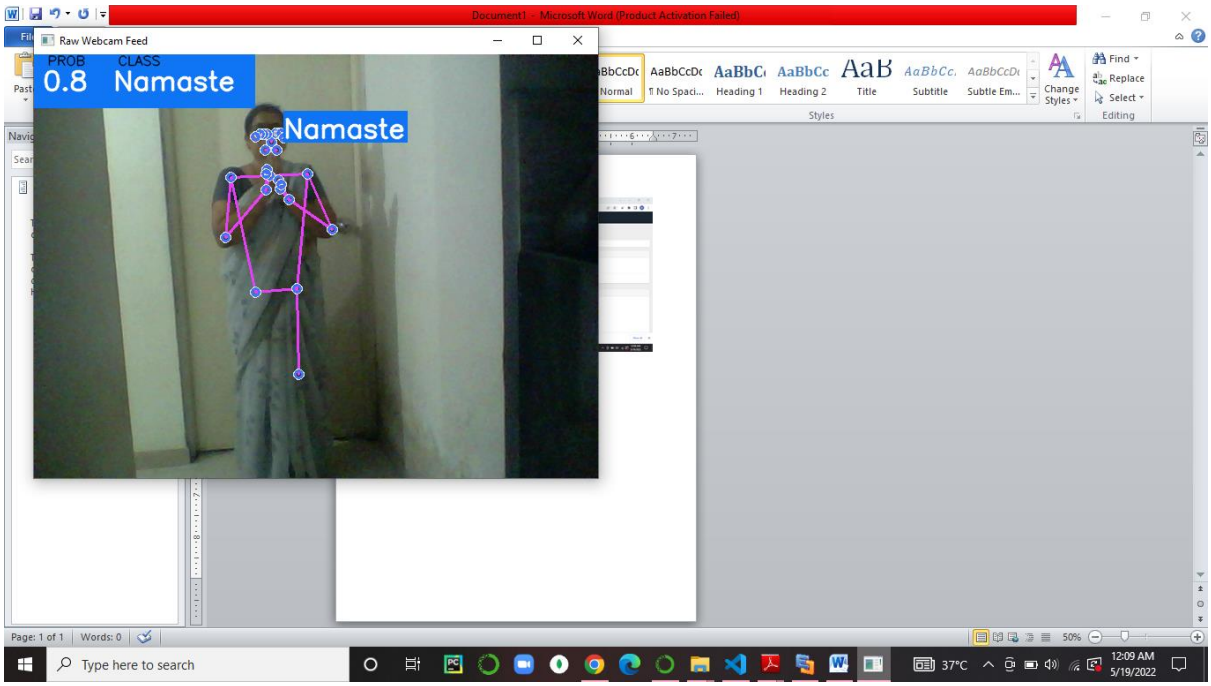


Figure 6.13 Live video Input (Namaste)

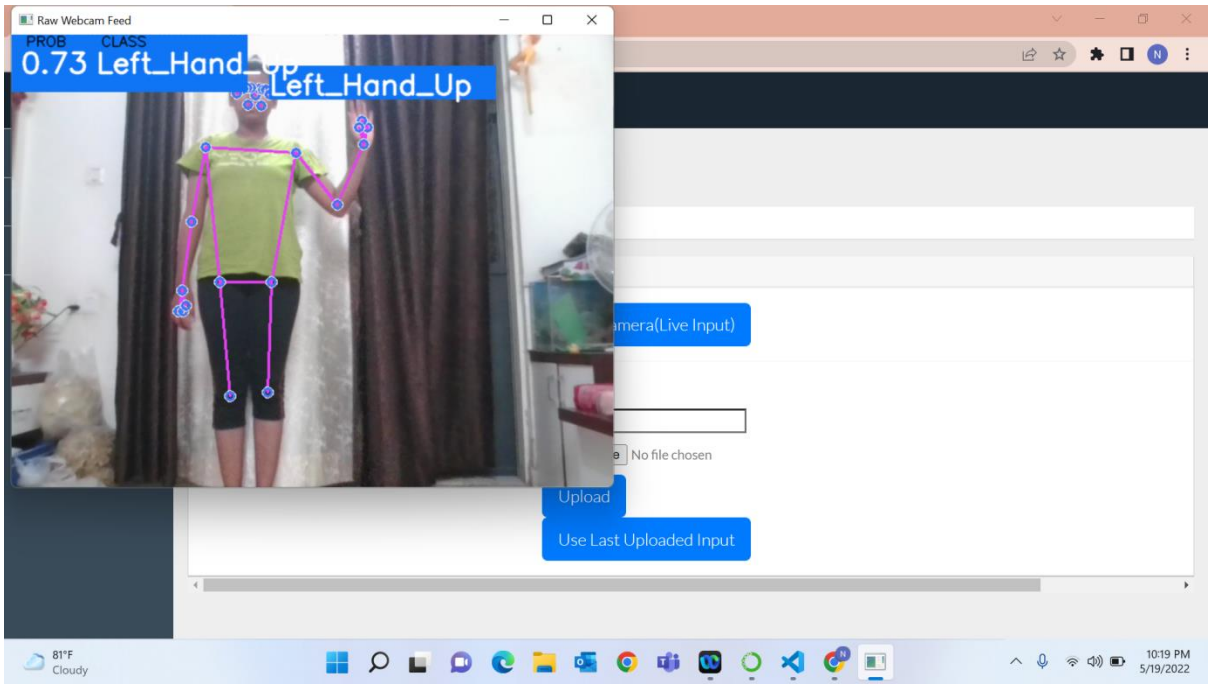


Figure 6.14 Live video Input (Left hand up)

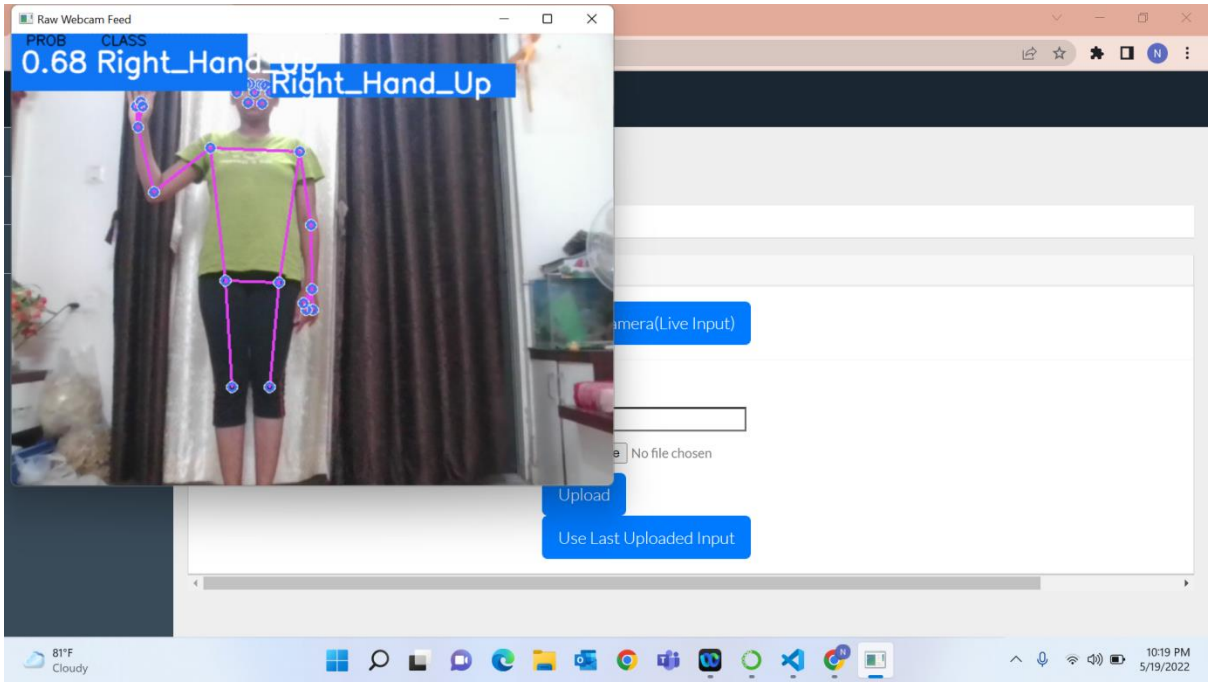


Figure 6.15 Live video Input (Right hand up)

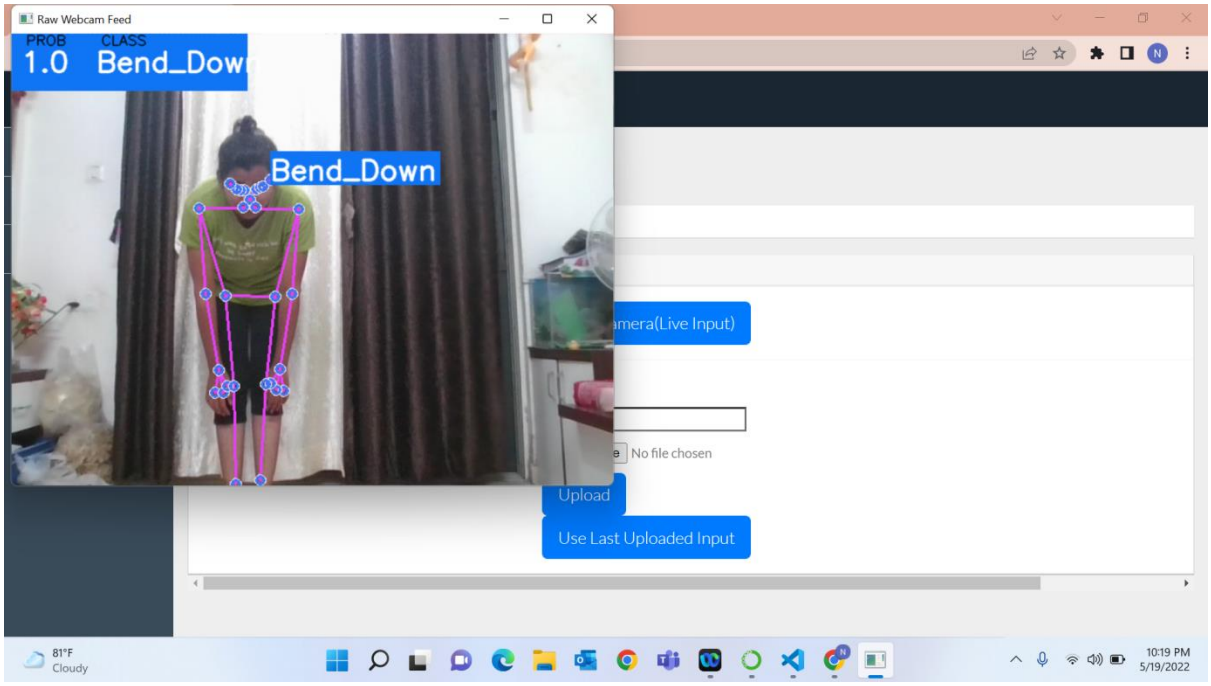


Figure 6.16 Live video Input (Bend down)

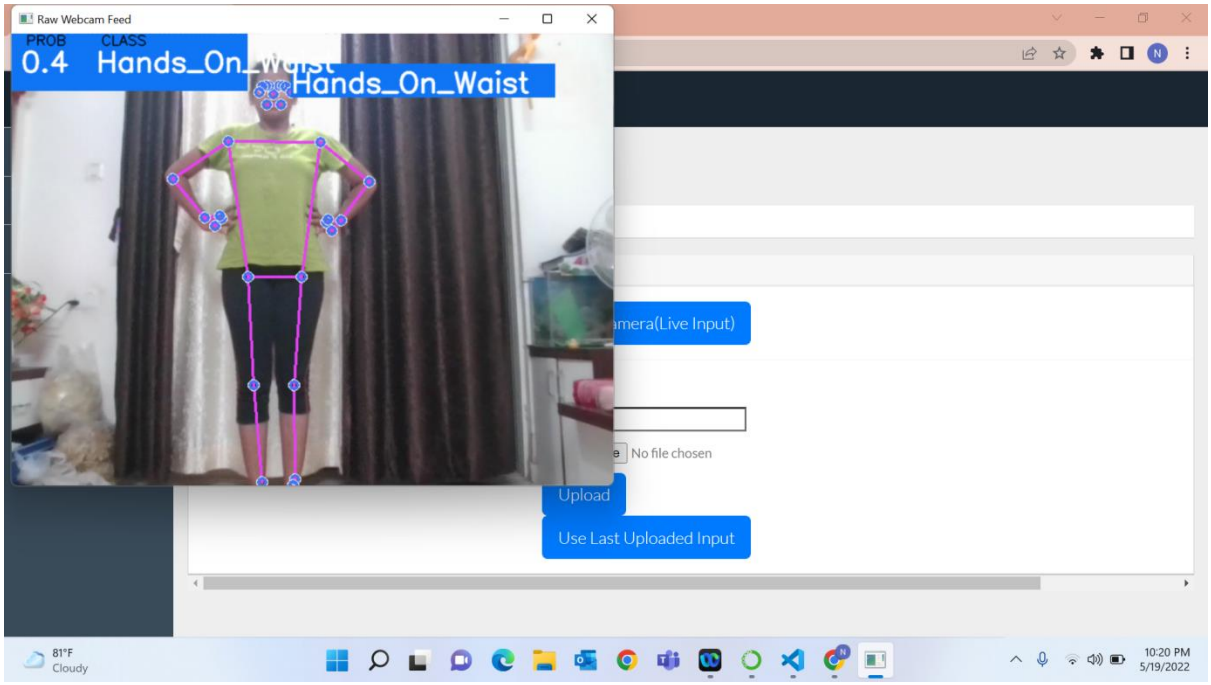


Figure 6.17 Live video input (Hands on Waist)

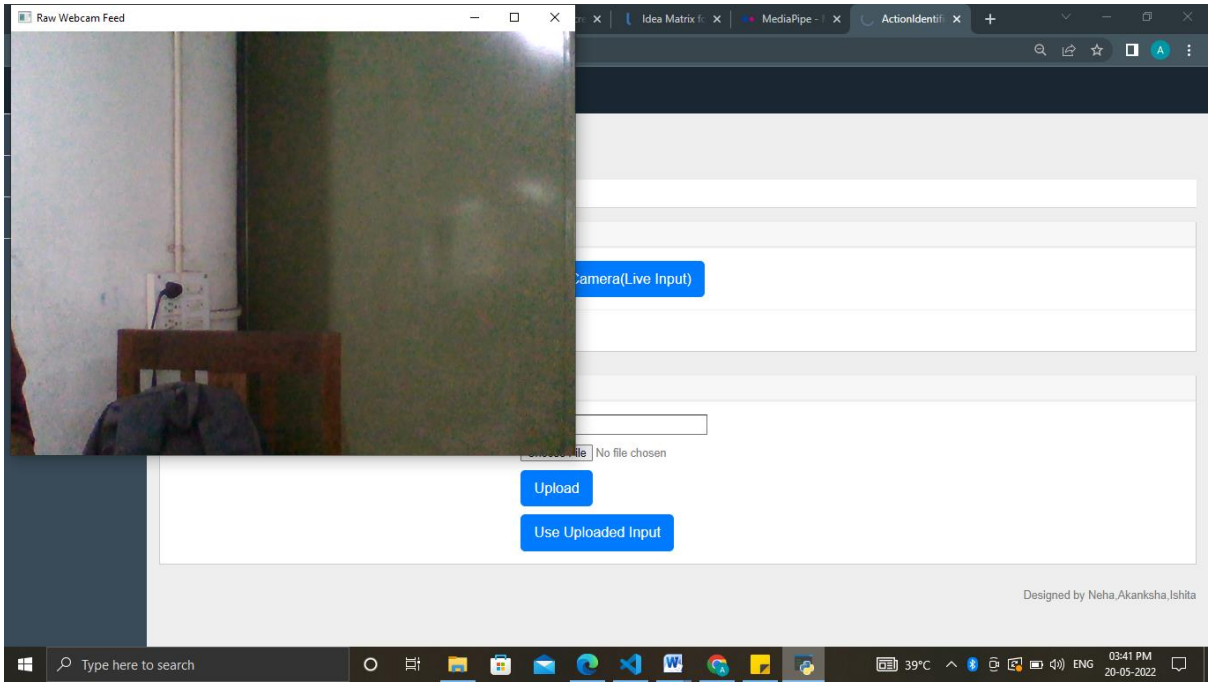


Figure 6.18 Live video input (no human identified)

6.3 Future Scope:

The project that we've implemented is suitable for identifying only single person in the frame. If there is more than one person it can only identify action of one person as shown below:

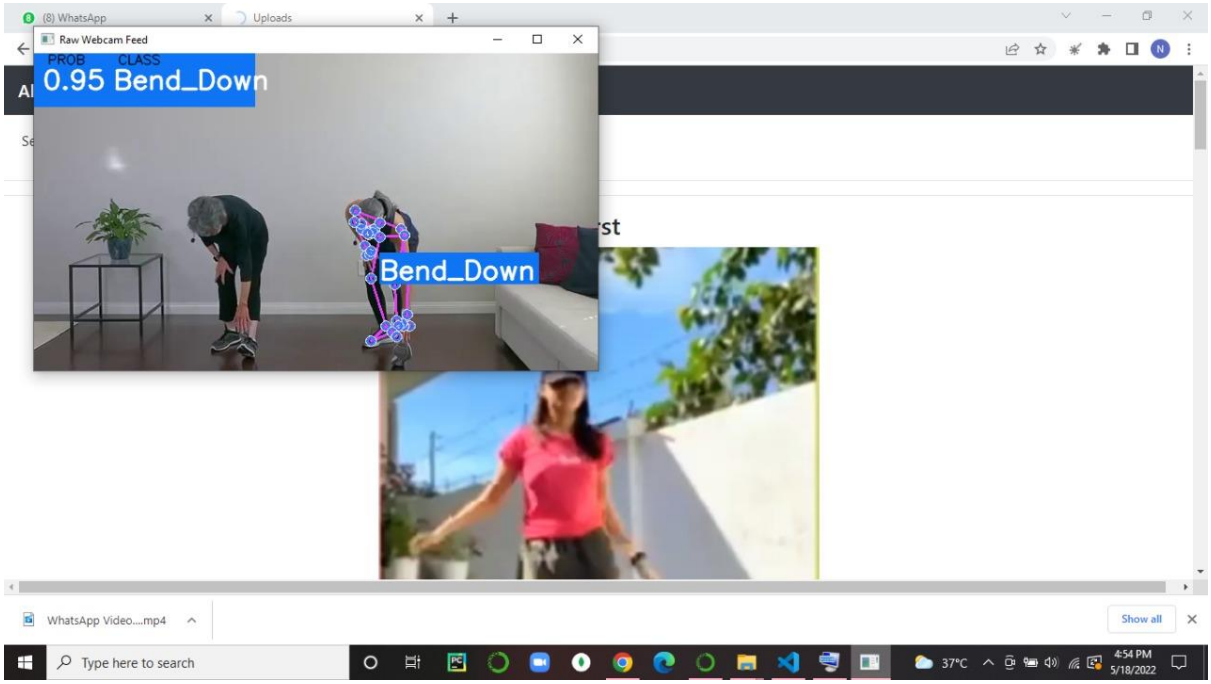


Figure 6.19 Multiple Person Activity Identification

In future we're planning to work upon this to identify multiple person activity in a single frame.

7: CONCLUSION

In this project we have introduced Human Activity Identification technique which will detect the activity being performed in the input video. We have proposed a Human Activity Identification system based on pose estimation and Classification algorithm. This system will combine the results of Pose Estimation model with the classification technique for better and more accurate detailed action prediction.

We have used Logistic regression, ridge classifier, random forest classifier, gradient boosting classifier and got 100% accuracy for Logistic regression, random forest classifier, gradient boosting classifier and 99% for ridge classifier. We used logistic regression as our final model for prediction

8: REFERENCES

[1] Aggarwal, J. K., & Cai, Q. (1999). Human motion analysis: A review. *Computer vision and image understanding*, *73*(3), 428-440.

[2] Kamel, A., Sheng, B., Yang, P., Li, P., Shen, R., & Feng, D. D. (2018). Deep convolutional neural networks for human action recognition using depth maps and postures. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, *49*(9), 1806-1819.

[3] Hossain, M. R. I., & Little, J. J. (2018). Exploiting temporal information for 3d human pose estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)* (pp. 68-84).

[4] H. P. S. A. Lateef and U. Eranna, "Modelling Approach to Select Potential Joints for Enhanced Action Identification of Human," 2018 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), 2018, pp. 1506-1509, doi: 10.1109/ICEECCOT43722.2018.9001313.

[5] R. A., Amiruzzaman, M., Ahmed, N., & Islam, M. R. (2020, August). Efficient Frequency Domain Feature Extraction Model using EPS and LDA for Human Activity Recognition. In *202020 3rd IEEE International Conference on Knowledge Innovation and Invention (ICKII)* (pp. 344-347). IEEE.

[6] Gupta, Abhay, Kuldeep Gupta, Kshama Gupta, and Kapil Gupta. "Human Activity Recognition Using Pose Estimation and Machine Learning Algorithm."

[7] Moeslund, T. B., Hilton, A., and Krüger, V. (2006). A survey of advances in vision-based human motion capture and analysis. Comput. Vis. Image Understand. 104, 90–126. doi:10.1016/j.cviu.2006.08.002

[8] Poppe, R. (2010). A survey on vision-based human action recognition. Image Vis. Comput. 28, 976–990. doi:10.1016/j.imavis.2009.11.014

[9] Chen, L., Wei, H., and Ferryman, J. (2013b). A survey of human motion analysis using depth imagery. Pattern Recognit. Lett. 34, 1995–2006. doi:10.1016/j.patrec.2013.02.006

[10] Aggarwal, J. K., and Xia, L. (2014). Human activity recognition from 3D data: a review. Pattern Recognit. Lett. 48, 70–80. doi:10.1016/j.patrec.2014.04.011

[11] Guo, G., and Lai, A. (2014). A survey on still image based human action recognition. Pattern Recognit. 47, 3343–3361. doi:10.1016/j.patcog.2014.04.018

[12] Pantic, M., and Rothkrantz, L. (2003). “Towards an affect-sensitive multimodal human-computer interaction,” in Proc. IEEE, Special Issue on Multimodal Human-Computer Interaction, Invited Paper, Vol. 91 (IEEE), 1370–1390.

[13] Pantic, M., Pentland, A., Nijholt, A., and Huang, T. (2006). “Human computing and machine understanding of human behavior: a survey,” in Proc. International Conference on Multimodal Interfaces (New York, NY), 239–248.

[14] Zeng, Z., Pantic, M., Roisman, G. I., and Huang, T. S. (2009). A survey of affect recognition methods: audio, visual, and spontaneous expressions. IEEE Trans. Pattern Anal. Mach. Intell. 31, 39–58. doi:10.1109/TPAMI.2008.52

[15] Bousmalis, K., Mehu, M., and Pantic, M. (2013a). Towards the automatic detection of spontaneous agreement and disagreement based on nonverbal behaviour: a survey of related cues, databases, and tools. Image Vis. Comput. 31, 203–221. doi:10.1016/j.imavis.2012.07.003

[16] Rodríguez, N. D., Cuéllar, M. P., Lilius, J., and Calvo-Flores, M. D. (2014). A survey on ontologies for human behavior recognition. ACM Comput. Surv. 46, 1–33. doi:10.1145/2523819